Monthly Notices of the ROYAL ASTRONOMICAL SOCIETY

MNRAS **526,** 1809–1821 (2023) https://doi.org/10.1093/mnras/stad2809
Advance Access publication 2023 September 29

An optimized transient detection pipeline for the ASKAP Variables and Slow Transients (VAST) survey

Tao An(安涛),[1,2,3]★ Baoqiang Lao(劳保强),[4] Zhijun Xu(徐志骏),[1,3]★ Shuoying Lu(卢铄迎),[5] Yuanming Wang(王远明),[6,7,8] Tara Murphy,[6,8] David L. Kaplan[9] and Shaoguang Guo(郭绍光)[1,2,3]

[1]*Shanghai Astronomical Observatory, CAS, 80 Nandan Road, Shanghai 200030, China*
[2]*School of Astronomy and Space Sciences, University of Chinese Academy of Sciences, No. 19A Yuquan Road, Beijing 100049, China*
[3]*Key Laboratory of Radio Astronomy and Technology, Chinese Academy of Sciences, A20 Datun Road, Chaoyang District, Beijing 100101, China*
[4]*School of Physics and Astronomy, Yunnan University, Kunming 650091, China*
[5]*Guilin University of Electronic Technology, Guilin 541004, China*
[6]*Sydney Institute for Astronomy, School of Physics, University of Sydney, Sydney, NSW 2006, Australia*
[7]*Australia Telescope National Facility, CSIRO, Space and Astronomy, PO Box 76, Epping, NSW 1710, Australia*
[8]*ARC Centre of Excellence for Gravitational Wave Discovery (OzGrav), PO Box 218, Hawthorn, Victoria, Australia*
[9]*Center for Gravitation, Cosmology, and Astrophysics, Department of Physics, University of Wisconsin-Milwaukee, P.O. Box 413, Milwaukee, WI 53201, USA*

Accepted 2023 September 12. Received 2023 September 5; in original form 2023 June 23

ABSTRACT

In this paper, we present an optimized version of the detection pipeline for the ASKAP Variables and Slow Transients (VAST) survey, offering significant performance improvement. The key to this optimization is the replacement of the original w-projection algorithm integrated in the Common Astronomy Software Applications package with the w-stacking algorithm implemented in the `WSClean` software. Our experiments demonstrate that this optimization improves the overall processing efficiency of the pipeline by approximately a factor of 3. Moreover, the residual images generated by the optimized pipeline exhibit lower noise levels and fewer artefact sources, suggesting that our optimized pipeline not only enhances detection accuracy but also improves imaging fidelity. This optimized VAST detection pipeline is integrated into the Data Activated Liu Graph Engine (`DALiuGE`) execution framework, specifically designed for SKA-scale big data processing. Experimental results show that the performance and scalability advantages of the pipeline using `DALiuGE` over traditional MPI or BASH techniques increase with the data size. In summary, the optimized transient detection pipeline significantly reduces runtime, increases operational efficiency, and decreases implementation costs, offering a practical optimization solution for other ASKAP imaging pipelines as well.

Key words: techniques: image processing – surveys – radio continuum: transients.

1 INTRODUCTION

The Universe is replete with highly variable objects that are often associated with extreme high-energy astrophysical phenomena. These objects provide a unique opportunity to study the high-energy universe in depth from both observational and theoretical perspectives. Many classes of transient sources have been discovered at different cosmological distances, including gamma-ray bursts and fast radio bursts, making them probes of the Universe all the way up to the epoch of cosmic reionization (e.g. Wijers et al. 1998; Gao, Li & Zhang 2014). Wide-field, high temporal resolution, and high sensitivity surveys span the entire electromagnetic spectrum from radio to TeV bands. Recent discoveries of gravitational waves (Abbott et al. 2016, 2017), extragalactic neutrinos (IceCube Collaboration 2018a, b), and high-energy cosmic rays (Cao et al. 2021) have opened new multimessenger windows, enriching our understanding of the Universe.

Two primary methods are typically used to discover and search for transient sources: the time series method and the image-domain method. A typical application of the time series method is to search for pulsars by observing periodic radio pulse signals. The image-domain method is used to identify suddenly brightening objects or previously unknown transient sources by comparing the difference between images taken at adjacent times or by subtracting from a reference image (e.g. Bond et al. 2001; Hurley-Walker et al. 2022).

Numerous large advanced telescopes at multiple wavelengths are conducting large field of view (FoV), high-sensitivity sky surveys to search for transient sources. These include the large-field high-cadence optical surveys such as the Zwicky Transient Facility (ZTF; Bellm et al. 2019) and the Large Synoptic Survey Telescope (LSST; Ivezić et al. 2019), as well as high angular resolution radio surveys conducted with the Very Large Array (VLA; Lacy et al. 2020), Australian SKA Pathfinder (ASKAP; Murphy et al. 2021), and MeerKAT (Fender et al. 2016). By cataloguing a large sample of celestial objects, these surveys will significantly contribute to discovering transient sources.

Rapid follow-up observations of newly discovered transient sources in images demand fast imaging capabilities, which require substantial data processing. Detecting transients from an image data base obtained from long-term observations involves processing massive amounts of data (e.g. Law et al. 2015). On the other hand,

★ E-mail: antao@shao.ac.cn (TA); xuthus@shao.ac.cn (ZX)

© 2023 The Author(s)
Published by Oxford University Press on behalf of Royal Astronomical Society

Downloaded from https://academic.oup.com/mnras/article/526/2/1809/7286492 by University of Wisconsin Milwaukee user on 30 September 2024

a comprehensive search for variable sources over a large sky area requires imaging, identifying, locating, and cataloguing all detectable objects within the FoV. When the first phase of the Square Kilometre Array (SKA) radio telescope (Dewdney et al. 2009) is completed, its sky survey will be about 50 times faster than the current largest radio telescope array. Consequently, the development of highly automated, highly reliable and fast imaging pipelines for detecting transient sources, based on the SKA precursor telescopes, has become crucial to meet the enormous challenge of the unprecedented amount of data (An 2019; Bonaldi et al. 2021) that the SKA will generate in the future.

The ASKAP (Hotan et al. 2021) is a new-generation survey radio telescope built in Western Australia at the same site as the SKA low-frequency array. ASKAP employs advanced technologies such as phased-array feeds (PAFs) to provide fast survey speeds and a wide FoV, making it the ideal instrument for untargeted searches for transient phenomena. The ASKAP Variables and Slow Transients (VAST; Murphy et al. 2013) is one of the key survey science projects of the ASKAP[1], and the scientific goals include the discovery and characterization of a wide range of transient and variable objects on time-scales from 10 s to 5 yr, including flaring stars, gamma-ray burst afterglows, intermittent pulsars, magnetars and extreme scattering events etc.

The VAST Collaboration has developed a pipeline for detecting transients and variables by analysing images obtained from the ASKAP imaging pipeline (Pintaldi et al. 2022). The transient detection pipeline of the VAST project extracts and measures information about the objects from the images, constructs light curves, and generates detection notification for subsequent analysis (Wang et al. 2023). The VAST Pilot Survey used all 36 antennas and was conducted between 2019 August and 2020 August, with a total of 162 h of observations and a total pilot survey area of 5131 square degrees (Murphy et al. 2021). The ongoing full VAST survey will focus more on performing faster imaging and transient source detection, requiring a more robust and automatic pipeline.

In this paper, we introduce the optimization of the VAST detection pipeline described in Wang et al. (2021), including replacing the w-projection imaging algorithm in the original pipeline with the w-stacking imaging algorithm, and integrating the optimized imaging software into the Data Activated *Liu* Graph Engine (`DALiuGE`), an intelligent execution framework designed for processing large astronomical data sets (Wu et al. 2017). These modifications are aimed at improving the execution efficiency of the pipeline. For ease of description, we refer to the original VAST transient detection pipeline as the'original pipeline' and the optimized transient detection pipeline as the 'optimized pipeline' in this paper. Section 2 describes the optimization method. Section 3 describes the integration of the VAST imaging pipeline into `DALiuGE`. Section 4 presents the summary.

2 OPTIMIZATION OF THE VAST IMAGING PIPELINE

ASKAP consists of 36 telescopes, each equipped with a phased array feed of 36 beams, capable of generating a substantial amount of raw data at an approximate rate of 100 Tb s^{-1}). These raw data are first correlated and integrated at the ASKAP observatory. The correlated visibility data are then exported at a rate of up to 2.4 GB s^{-1} and transferred to the Pawsey Supercomputing Centre in Perth, Western Australia, for subsequent data processing and storage (Hotan et al. 2021). The ASKAP VAST Pilot Survey was carried out during 2019–2020, accumulating 162 h of observations in total (Murphy et al. 2021). The full survey operation of the ASKAP started in late 2022. The VAST Pilot Survey has validated the observation strategies, data processing capabilities, and scientific analysis methodologies for the VAST project, paving the way for the full-scale sky surveys.

2.1 Transient detection pipeline used in the VAST pilot survey

The transient detection pipeline used in this work was originally developed by the VAST Collaboration (Wang et al. 2021). Data from each of the 36 ASKAP beams are processed independently. The original pipeline consists of several key steps: ingesting calibrated visibility data, fixing the observation phase centre, making deep sky models, subtracting the models from visibility data, generating snapshot images, estimating background and noise levels, producing deep image cutouts, detecting sources, creating light curves, and identifying candidates of variables or transients etc. The flow chart of the whole pipeline is shown in Fig. 1 (see also fig. 3 in Wang et al. 2023). Wang et al. (2023) aim to present the VAST transient detection pipeline and the scientific results of the pilot survey, so their flowchart details the procedure from the online processing of the raw ASKAP visibility, then to the data processing using the transient detection pipeline deployed at the China SKA Regional Centre to generate transient candidates, and finally to produce a transient catalogue after manual checks. Our research, on the other hand, focuses on the optimization of the detection pipeline in the middle phase, which does not include the initial online processing and the final manual checks, as seen in Fig. 1. The main steps are outlined below.

2.1.1 Ingesting calibrated visibility data

The input data to the VAST imaging pipeline come from the CSIRO ASKAP Science Data Archive (CASDA; Chapman et al. 2017) and have been pre-calibrated using the ASKAP pipeline (Guzman et al. 2019) at the Australian Supercomputing Centre (Pawsey). The pre-calibration includes the calibration of phase errors introduced in the signal by the observing equipment and the atmosphere, the removal of radio frequency interference (RFI), and the calibration of each antenna bandpass.

2.1.2 Fixing the observation phase centre

ASKAP maintains a phase centre for each beam (Hotan et al. 2021), which means that it has 36 phase tracking systems. To ensure astrometric performance, ASKAP records the offset of each beam's phase centre during the observation in an MS-format file from which the imaging pipeline can obtain the corresponding beam offset information and correct the phase centre prior to the imaging procedure.

2.1.3 Making a deep CLEAN model and subtracting the model from visibility data

This step is mainly done using the *tclean* task in the Common Astronomy Software Applications (`CASA`) software package (McMullin et al. 2007).

ASKAP has a large FoV of 30 square degrees, so the wide-field effect must be calibrated. The original pipeline uses the w-projection imaging algorithm (Cornwell, Golap & Bhatnagar 2005) to handle

[1] https://www.atnf.csiro.au/projects/askap/ssps.html

Downloaded from https://academic.oup.com/mnras/article/526/2/1809/7286492 by University of Wisconsin Milwaukee user on 30 September 2024

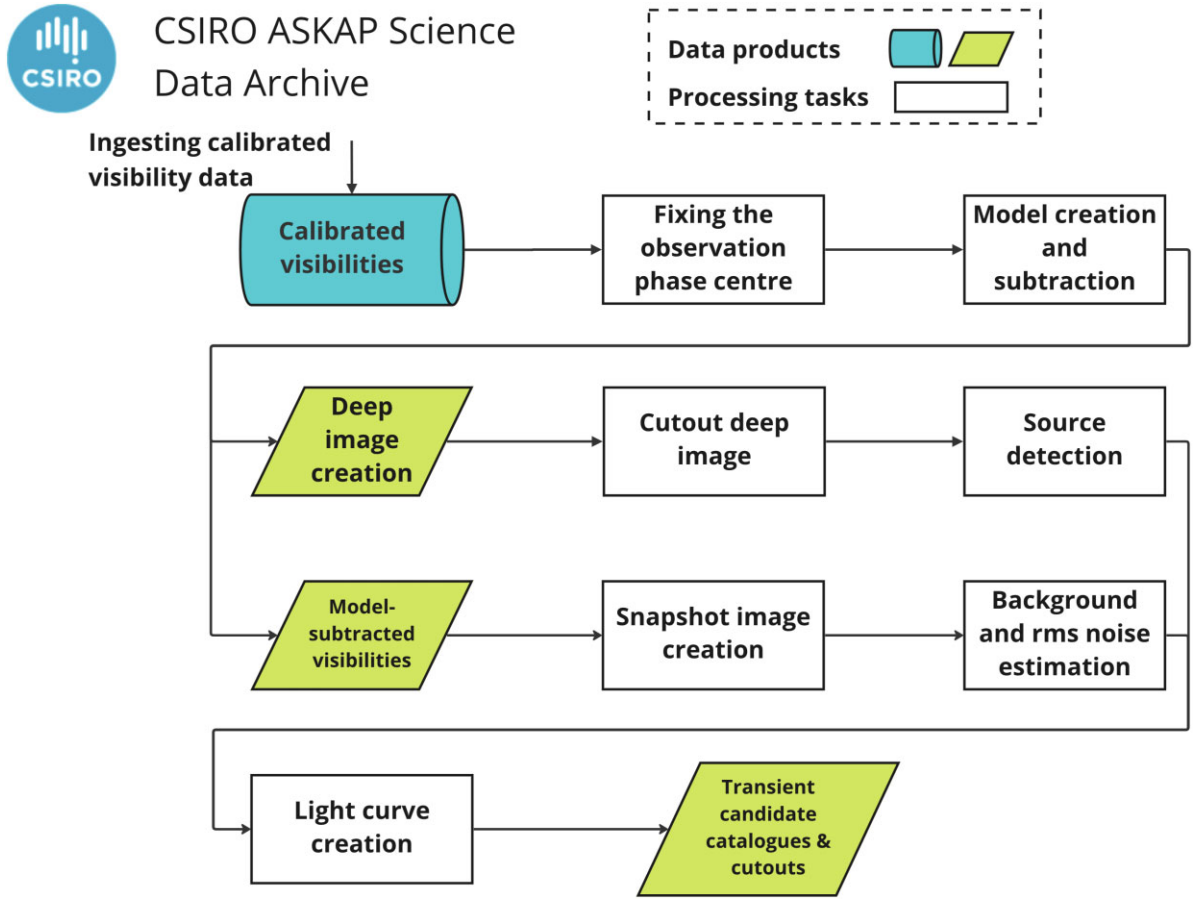

Figure 1. Flowchart of our optimized transient detection pipeline, specifically designed for processing and analysing VAST data. While Wang et al. (2021) described the entire process from the online processing of the ASKAP raw visibility to the transient detection pipeline and finally to the final transient catalogue, our flowchart concentrates on the central stage of the data processing and analysis that are the focus of our optimization efforts.

the bias caused by the antenna baseline being projected on different planes during imaging.

The large FoV of the ASKAP contains radio sources of multiple sizes, including compact point-like sources and large-scale extended radio galaxies and galaxy clusters. However, the traditional CLEAN algorithm, which decomposes the sky image into a collection of point sources or scaled delta functions, is no longer adequate for imaging objects with complex structures. Therefore, the pipeline employs the Multi-Scale CLEAN approach (Cornwell 2008) during the iterative CLEAN process. When the *tclean* task is finished, the final CLEAN model will be stored with an MS file. Next, the CLEAN model is subtracted from the MS data by using the `CASA` task *uvsub* to obtain the residual data.

In our optimized pipeline, the `WSClean` program is used instead of the `CASA` package to perform the imaging operation, and we found that `WSClean` based on the w-stacking algorithm is faster than `CASA` with the w-projection algorithm in this transient detection pipeline, a result that is consistent with the general expectation in wide-field radio imaging (see Offringa et al. 2014). More discussion is given in Section 2.2.

2.1.4 *Snapshot image creation*

Imaging the entire continuous hours of data is a huge challenge for the current computing system. A practical operation is to slice the residual visibility data generated in the previous step in a time sequence in order to meet the need for fast imaging and fast scientific output. For example, 7-h observational data are divided into 28 segments of 15 min of residual data and imaged separately. The sliced data segments can be imaged independently and run in parallel on a multinode, multicore supercomputing system, where processing is much faster.

2.1.5 *Background and rms noise estimation*

This is performed on the residual snapshot images using *Aegean* software (Hancock, Trott & Hurley-Walker 2018), and the results will be used in subsequent steps for variable source identification and light-curve construction.

2.1.6 *Generating cutout deep image*

To ensure that subsequent analyses are performed within an effective central FoV, a central image of 3000 pixel × 3000 pixel is cropped around the centre of the original deep image of 10 000 pixel × 10 000 pixel. The current sky model derived from the shallow ASKAP-RACS survey (Hale et al. 2021; Duchesne et al. 2023) is not yet sufficient for accurate calibration of the data and use as a reference image. It is therefore necessary to carry out deep imaging in the current VAST pipeline. In the future, when full-sky deep ASKAP surveys such as Evolutionary Map of the Universe (EMU;

Downloaded from https://academic.oup.com/mnras/article/526/2/1809/7286492 by University of Wisconsin Milwaukee user on 30 September 2024

Norris et al. 2021) and Magnetic Polarisation Survey of the Universe (POSSUM; Anderson et al. 2021) are completed, it will be possible to build a sufficiently accurate sky model to use as a reference image. This will allow us to perform transient detection by subtracting the reference image directly from the VAST snapshot images, without the need for deep imaging. The latter approach can greatly improve the efficiency of the transient detection, but it is dependent on the accuracy of the sky model. Direct subtraction of the reference image can introduce errors that affect the accuracy of transient detection if the sky model is not accurate enough.

2.1.7 *Source detection*

The search for radio sources in the generated deep images is performed using the *Aegean* software. A deep image catalogue is generated.

2.1.8 *Light curve creation*

The light curve of each source is extracted from the deep image catalogue following three steps (Wang et al. 2021, 2023). First, the peak flux density $S_{\rm deep}$ and fitted source position are obtained from the deep image catalogue. The flux density $S_{\rm deep}$ at the ith pixel is obtained by forcing the measurement of the flux density at the peak position of the source in the deep sky image, rather than as a result of a free-fit, which avoids the bias of the measured flux density due to positional deviations. Secondly, the peak flux density $S_{i,\rm diff}$ and rms noise σ_i in the ith residual image at the corresponding fitted position are measured. Thirdly, the final peak flux density S_i for each data point used to construct the light curve is given by $S_i = S_{\rm deep} + S_{i,\rm diff}$. Each beam data will generate $N_{\rm s}$ light curves, where $N_{\rm s}$ denotes the number of detected sources. Next, all light curve results are used as inputs for variability analysis. The modulation index $m = \sigma_s/\bar{S}$ is used to describe the magnitude of variability of radio sources quantitatively (Murphy et al. 2021), where σ_s is the standard deviation of flux density of the light curve, and $\bar{S}$ is the weighted mean flux density. The chi-squared value $\chi^2_{\rm lc}$ is used to measure the significance of the random variance for light curves, calculated as follows:

$$\chi^2_{\rm lc} = \sum_{i=1}^{N_{\rm t}} \frac{(S_i - \bar{S})^2}{\sigma_i^2}. \quad (1)$$

Under the null hypothesis, χ^2_{lc} follows the theoretical chi-squared distribution with $N_{\rm t} - 1$ degrees of freedom, i.e. $\tilde{\chi}^2_{\rm lc} = \frac{\chi^2_{\rm lc}}{N_{\rm t}-1}$. The variation probability $P(\chi^2_{lc})$ for each light curve is calculated by cumulative distribution function (CDF) of theoretical chi-squared distribution (Bell et al. 2014; Wang et al. 2021).

2.1.9 *Creating transient candidates catalogue and cutout images*

The final transient candidates are identified based on the following four conditions: the value of $\tilde{\chi}^2_{\rm lc}$ is higher than 3σ; the modulation index m exceeds 3 per cent; the astrometric position of source is less than 0.8 deg from the beam centre; the compactness $S_{\rm int\_flux}/S_{\rm peak\_flux} < 1.5$, where $S_{\rm int\_flux}$ and $S_{\rm peak\_flux}$ are the integrated flux density and the peak flux density of candidate, respectively. The last condition ensures that the variable sources are compact and not caused by image artefacts or sidelobes of bright sources. Finally, the information on the final candidates is saved in a CSV file and the light curves are plotted. The residual 8-arcmin snapshot images are cropped and combined in time sequence into a'.gif' file. The above conditions do not fully and automatically exclude all false detections, such as misidentified candidates in the sidelobes of a bright source. These spurious candidates must be verified by further visual inspection.

2.2 Improvement of the pipeline

Wide FoV imaging of radio interferometric observations is a computationally intensive task, especially for the large amounts of data that are generated by modern non-coplanar arrays. Offringa et al. (2014) developed a fast wide-field imaging and deconvolution algorithm, w-stacking Clean (`WSClean`). In their imaging experiments using the Murchison Widefield Array (MWA) data, the `WSClean` was an order of magnitude faster than the *tclean* with w-projection algorithm integrated in `CASA` (Offringa et al. 2014). The image quality derived from w-stacking is comparable to that w-projection, and in some cases slightly better. These differences are due to the differences between the two algorithms themselves. The w-projection algorithm in *tclean* involves first convolving the visibility data in the uv plane with a function that depends on the w term and then performing an inverse Fourier transform. This convolution operation is computationally expensive, especially when the number of w-planes is large. In contrast, w-stacking algorithm divides the visibility data into different layers based on the w terms and then the inverse Fourier transform is applied to each layer separately. The results are then summed to produce the final image. This process is more computationally efficient because the convolution operation associated with w-projection is replaced by the simpler stacking operation. Therefore, as long as the number of w-planes (and the computational complexity of the convolution operation) is large, `WSClean` can be expected to be faster than algorithms based on w-projection.

It is important to note that the two techniques have different advantages and trade-offs. w-stacking algorithm is faster and more memory efficient, but it can only correct the w terms within a given FoV. Although w-projection is more computationally expensive, it corrects w terms in the entire FoV. Furthermore, from a workflow execution point of view, w-stacking is faster when visibility gridding is the main cost of the algorithm. On the other hand, the w-projection algorithm is faster when the inverse FFT is the main cost of the workflow. The exact performance difference depends on the specifics of the data set and the imaging requirements. Therefore, when choosing between these two methods, practical considerations (e.g. computational resources) and specific scientific goals of the imaging task should be taken into account. Furthermore, in our experiments, we found that the `CASA` software needs to load a large number of startup items before execution. In this case, if multiple nodes are running in parallel, individual nodes may get stuck at startup, which limits the operational speed of the CASA pipeline when processing large amounts of data.

We first examine the runtime for each step of the original pipeline (shown in Fig. 2). The most time-consuming step is making a deep CLEAN model and subtracting it from the visibility data (the 'model' step), accounting for 93.98 per cent of the total runtime. In particular, almost all of the time is spent on the *tclean* task. The w-projection algorithm in the *tclean* task first performs imaging by convolving visibilities with different discrete w-kernels and then performs an FFT on the results. The size of the w-kernel directly affects the speed of the algorithm, as most of the running time of the w-projection is consumed by the convolution operation. Since the ASKAP data have large w-values ($>20\,000\lambda$), the w-projection algorithm will use very

Downloaded from https://academic.oup.com/mnras/article/526/2/1809/7286492 by University of Wisconsin Milwaukee user on 30 September 2024

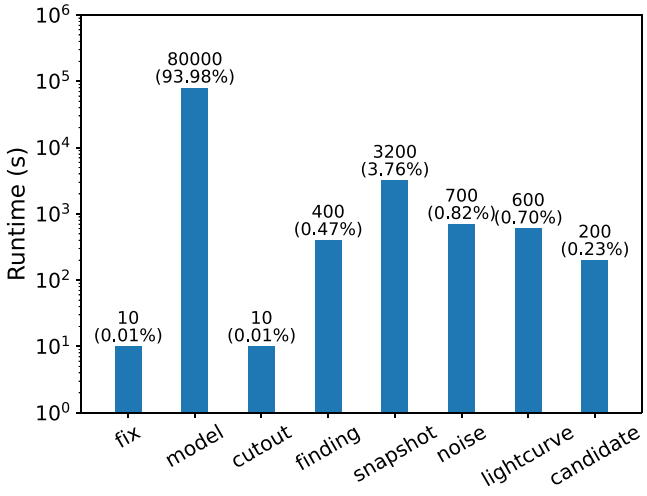

Figure 2. A demonstration of typical runtimes for each step of the original VAST transient detection pipeline. The labels 'fix', 'model', 'cutout', 'finding', 'snapshot', 'noise', 'light curve', and 'candidate' represent the steps of fixing the observation phase centre, making a deep CLEAN model, subtracting the model from visibility data, snapshot image creation, generating cutout deep image, source detection, background and rms noise estimation, light-curve creation, and identifying candidate variable sources, respectively.

large w-kernels (e.g. >688 pixels), making it very slow. In contrast, the w-stacking algorithm in `WSClean` first obtains images of each w-layer by executing an FFT on the gridded visibilities, then each image is multiplied by the w-projection term and the results are added together to obtain the final image. Since the convolution operation is replaced by multiplication, w-stacking is generally faster than w-projection, especially for large w values in this experiment. Hence, we use `WSClean` instead of the *tclean* task in `CASA` for imaging and deconvolution in our optimized pipeline. Furthermore, `WSClean` implements and optimizes multi-scale CLEAN and multifrequency deconvolution, allowing for mapping both the compact and diffuse emission sources (Offringa & Smirnov 2017). Specifically, for the wide-band imaging in this work, the minor loop of the w-stacking method is 2–3 orders of magnitude faster than multi-scale CLEAN and multifrequency w-projection CLEAN in *tclean*.

Table 1 lists the parameter settings of one of the experiments in this paper, including output image size, pixel size, weighting mode, and the data column parameters. We used multiscale CLEAN (see discussion above) by setting a parameter '-multiscale' with scales of '0, 5, 10, 15, 25'. The gain of the major CLEAN iterations is set to 0.85. During multiscale CLEAN, an automatic mask with 3σ allows structures below the noise to be CLEANed. The joined-channel CLEAN was enabled by setting '-channels-out' to 4, and combined with '-join-channels' to generate the multifrequency synthesis image. To get the deep images, the maximum number of CLEAN iterations is set to a large value, i.e. 100 000. When the '-auto-threshold' option is enabled, CLEAN stop threshold uses the automatic σ method and automatically sets the CLEAN threshold for each major cycle based on the noise level in the residual image.

The second most time-consuming step is the generation of snapshot images (3.76 per cent of the total runtime). It is responsible for imaging each snapshot residual data based on the start and end indices. In this step, there is a loop operation for sequentially processing N_t residual snapshot data. As each snapshot is imaged independently, this process can be optimized in parallel using Python *multiprocessing* technique. To optimize, we rewrote the code involving *tclean* steps using `WSClean` and PYTHON Casacore (van Diepen 2015), as well as performing parallel optimization of the generating snapshot images and noise estimation steps using PYTHON multiprocessing module. We also use the same approach to optimize the parallel steps for background and rms noise estimation. Specially, the residual snapshot data are equally divided into N_p portions and N_p CPU processes are set to execute the imaging in parallel.

We have also improved the generality of the deep imaging step. Signals entering the sidelobes of the primary beam, with an FoV of 30 square degrees at 800 MHz, can affect the signal detected in the primary beam. This sidelobe effect is most severe when a bright extended source ($\geq$50 Jy) falls into the sidelobes (e.g. Wang et al. 2021) and may greatly affect the accuracy of the transient source search. However, in many of the sky regions investigated, the sidelobes of the primary beam do not detect a bright source or the flux density of the detected bright source is not significant, so there is no need to set the output image with a very large pixel size for these data. To improve the generality of the pipeline and reduce unnecessary calculation time, we proposed two solutions: (1) uniformly set a smaller image size of 3000 pixel $\times$ 3000 pixel and determine whether there is a source brighter than 50 Jy outside the FoV before deep imaging. Then, the Peeling algorithm (Williams et al. 2019) is used to remove this bright source from the calibration data model before imaging. (2) Before deep imaging, check whether there is a bright source $\geq$50 Jy within the larger pixel size (i.e. 10 000 pixel) range of the data phase centre using the catalogue of the ASKAP continuum survey. Otherwise, a smaller pixel size is used. After comparing the operational efficiency of these two methods, we chose the latter.

2.3 Validation experiments and analysis

The following validation experiments were performed on an X86 compute node of the China SKA Regional Centre Prototype (CSRC-P; An, Wu & Hong 2019; An et al. 2022), which has two Intel Xeon Gold 5218 2.3-GHz processor with 32 CPU cores, 768 GB RAM memory and 1.2-TB local hard disc. Our test data were selected from the 29th beam data of the observation ID # 9602 (SB9602_beam29) in the ASKAP Pilot Survey for Gravitational Wave Counterparts (Wang et al. 2023). This observation started at UT 14:11:22 on 2019 August 16 and lasted 10.5 h.

Fig. 3 compares the runtimes of the original and the optimized pipelines on the X86 compute node. From the experiments running deep imaging and model building of SB9602_beam29 data (step #2 'model'), the original pipeline took about 22.1 h, while the optimized pipeline took about 8.3 h, which is about 2.7 times faster than the former. Moreover, the runtimes of the optimized pipeline in step #5 ('snapshot' image creation) and step #6 ('noise' estimation) are $\sim$1/11 of those of the original pipeline. The other steps of the optimized pipeline are slightly faster than the original pipeline. The total runtimes of the optimized pipeline and original pipeline are 31 320 and 84 318 s, respectively. Therefore, we can reduce the runtime by a factor of 2.7 using the optimized pipeline compared to the original pipeline.

In order to verify the validity of our improved method, the deep images obtained by both pipelines and their final detection results are compared and analysed below. The image quality information of cutout deep images by both pipelines is shown in Table 2. The mean *rms* noise obtained by the optimized pipeline is 26.0 μJy beam^{-1}, while the original pipeline obtained an *rms* noise of 28.4 μJy beam^{-1}, with the former showing a 9 per cent improvement over the latter. A slightly larger beam was obtained by `WSClean` due to the different

Downloaded from https://academic.oup.com/mnras/article/526/2/1809/7286492 by University of Wisconsin Milwaukee user on 30 September 2024

Table 1. WSClean imaging parameters.

Parameters	Settings
Image size	10 000 pixel × 10 000 pixel
Pixel size	2.5 arcsec
Maximum CLEAN iterations	100 000
Automatic threshold	σ
Automatic mask	3σ
Gain for major CLEAN loop	0.85
Imaging weighting	'Briggs' mode (robust parameter = 0.53)
'-Channels-out'	Four sub-channels jointly cleaned
Deconvolution mode	Multiple scales of '0, 5, 10, 15, 25'

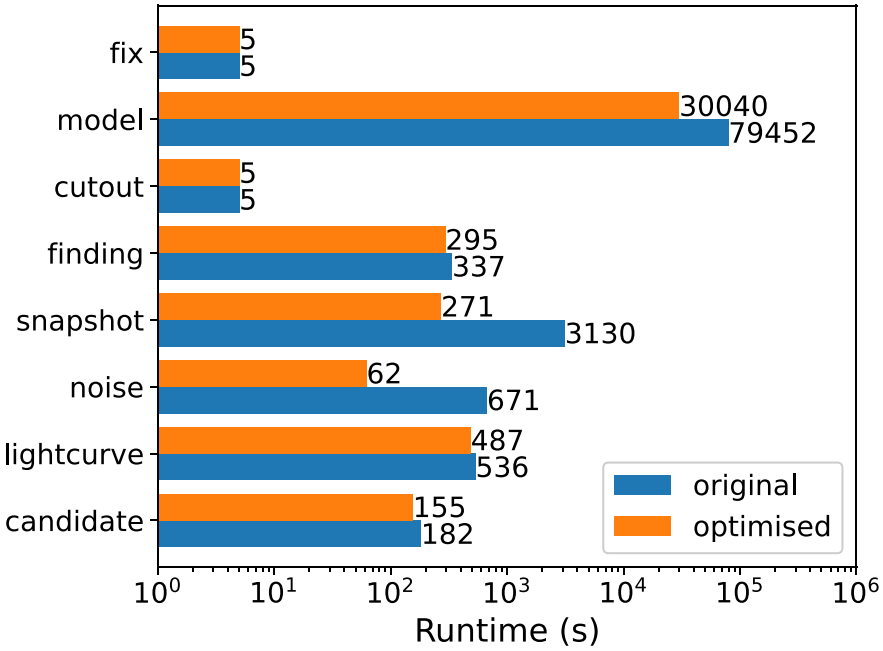

Figure 3. Comparison of the runtimes between the original pipeline and the optimized pipeline on the X86 compute node. The runtime of each step is shown in the left-hand panel, and the labels of 'fix', 'model', 'cutout', 'finding', 'snapshot', 'noise', 'light curve', and 'candidate' represent the steps fixing the observation phase centre, making a deep CLEAN model and subtracting the model from visibility data, generating snapshot images, cutout deep image, source finding, background and *rms* noise estimation, extracting light curves, and candidates selection and plots, respectively.

Table 2. Comparison of cutout deep image quality between the original and optimized pipelines on SB9602_beam29.

Methods	Mean rms noise (μJy beam^{-1})	Beam (maj × min, PA)	No. of detected sources
Original pipeline	28.44	17.15 arcsec × 13.65 arcsec, 80.59 deg	1695
Optimised pipeline	25.98	18.53 arcsec × 16.01 arcsec, 95.08 deg	1686

Briggs weighting parameters used in the two pipelines. The obtained cutout deep images were searched separately for radio sources using *Aegean* software, and the catalogues were exported. Other source-finding software and method can also be used, for example, deep learning-based `HeTu` (Lao et al. 2021). In the `CASA` image, 1695 radio sources are detected, and in the `WSClean` image, 1686 radio sources are detected.

Fig. 4 shows an example comparing the residual deep images obtained by the two pipelines. The images show the central 200 pixel × 200 pixel region. The residual deep image of the optimized pipeline has a much smoother noise distribution than the residual image obtained by the original pipeline. Some residual emission from the model-extracted radio sources is obviously visible in the original pipeline's output image, but not in the optimized pipeline's output image, indicating that the optimized pipeline produces higher quality images, deeper CLEANing and more accurate source extraction.

The choice between `WSClean` (w-stacking) and *tclean* (w-projection) depends on the specific requirements of the data and the scientific objectives. If the data have significant wide-field effects, w-stacking may be preferred due to its greater ability to handle w terms. However, if the data require more sophisticated deconvolution techniques, or if there are other factors at play, then w-projection may be preferred. However, how much better `WSClean` (w-stacking) can be than *tclean* (w-projection) needs to be quantified in the context of the specific application, as it depends on a number of factors such as field of view, array configuration, signal-to-noise ratio, dynamic range and specific scientific objectives. The effectiveness of these calibration and imaging techniques varies depending on the specific

Downloaded from https://academic.oup.com/mnras/article/526/2/1809/7286492 by University of Wisconsin Milwaukee user on 30 September 2024

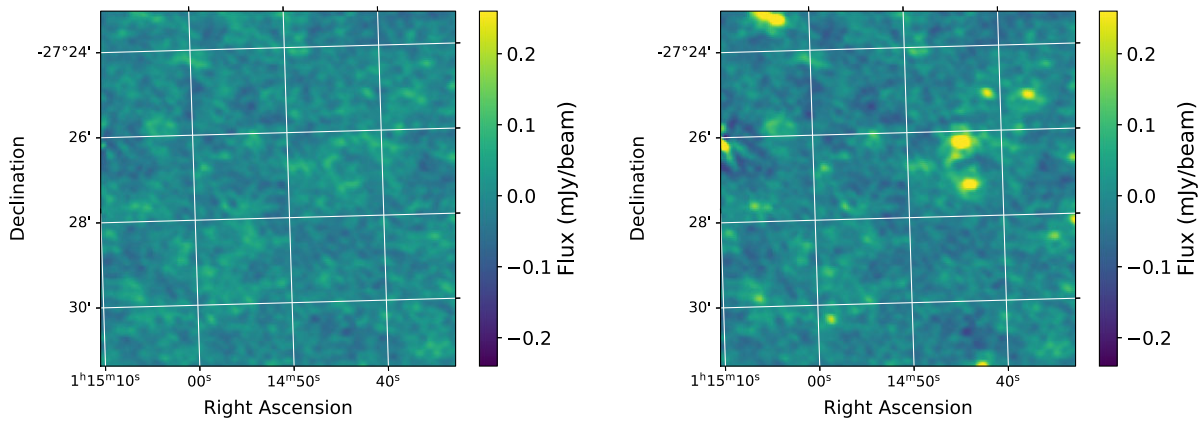

Figure 4. Residual deep image comparison between the optimized (left) and original pipelines (right). The residual map in the left-hand panel shows lower rms noise and less artefacts than the right-hand panel map.

observing conditions and science objectives. To better understand their relative performance, it would be desirable to systematically compare these techniques under a variety of conditions (e.g. different combinations of parameter settings), not only to greatly improve our understanding of the techniques, but also to gain insight into their applicability to different types of data and science cases, but this would require a large number of experiments.

For further analysis, the catalogues obtained from the two pipelines were cross-matched and compared using the Tool for OPerations on Catalogues And Tables (TOPCAT; Taylor 2005). Within 10 arcsec, there are 1513 matched sources, accounting for 90 per cent of the total number of sources detected. We plotted histograms of ΔRA, ΔDec., and the peak flux density of the matched sources and then fitted the histograms with Gaussian functions in Fig. 5. The differential RA and Dec. distributions of the cross-matched objects are close to zero: the peak distribution of ΔRA is at $+0.025$ arcsec and the peak distribution of ΔDec. is at $+0.0085$ arcsec. The standard deviations are very small, 0.27 arcsec for ΔRA and 0.19 arcsec for ΔDec., respectively, indicating that the catalogue obtained by the optimized pipeline generally agrees with that obtained by the original pipeline. The mean value of the peak flux density ratio distribution is close to 1 (with a peak at 1.01), and the standard deviation is 0.049.

After candidate selection, we found six variable source candidates from the optimized pipeline and nine from the original pipeline, five of which have matched locations. The information of the matched candidates is shown in Table 3. Four variable sources have been identified as transients in the previous study by Wang et al. (2021): J005800.93−235449.08, J005806.74−234744.51, J005808.98−233453.97, and J005812.02−233735.64 (i.e. the first 4 rows in Table 3). The cutout deep images and light curves of true transients by the optimized pipeline are shown in Figs 6 and 7, respectively. Upon visual inspection, other candidates are determined to be false transient detections (Wang et al. 2023).

Fig. 9 displays images obtained using the original pipeline from the test data of B9602_beam29, showing strong sidelobes around the bright central sources. The level and shape of sidelobes changed at different observing times due to differences in (u, v) coverage, and the original pipeline identified one of the sidelobes (the red-coloured cross) as a variable source candidate. In contrast, the optimized pipeline obtained lower and more uniformly distributed noise in the deep images as well as deeper CLEAN models (Table 2 and Fig. 4), which not only enables the detection of weak sources but also prevents to some extent the sidelobes from being misidentified as variable source candidates.

As depicted in Figs 8 and 9, J010021.98−232630.86 and J010117.82–234312.78 represent two false sources centred on the bright source but not true detection. Given their relative distance from the centre of beam #29 and closer to beam #35, neither source appeared in the detection of beam #35. This indicates that false detection can also occur when sources are located far from the centre of the beam. As an example, the three pseudo sources shown in Fig. 4 were not identified in the optimized pipeline. Moreover, the optimized pipeline obtained higher $\tilde{\chi}^2_{\rm lc}$ and m values of the true transients than for the original pipeline. Therefore the confidence level of the variable candidates detected by the optimized pipeline is higher than that of the original under the same metrics.

In this experiment, the fraction of pseudo-sources in the direct output of the optimized pipeline in this experiment is 1/3, whereas in the results from the original pipeline, this fraction is higher (more than 50 per cent). The reason for fewer false transient detection of the optimized pipeline can be attributed to its use of more advanced wide-field imaging and CLEAN algorithms which produce lower level and smoother image noise. As a result, spurious sources in the images, particularly those located in the sidelobes of bright sources, are substantially suppressed. However, it is important to note that the number or proportion of spurious sources in the direct output of the imaging pipeline is affected by a variety of observational and data-processing factors, including the treatment of w terms (e.g. the selection of different wide-field imaging algorithms in this study), CLEAN threshold settings, imaging weights, calibration errors, and unflagged RFI. Only by analysing these factors together can a more quantitative understanding of the proportion of pseudo sources in the pipeline output be obtained, and the pipeline needs to be run with different parameter configurations on different data sets.

3 INTEGRATION OF THE TRANSIENT DETECTION PIPELINE ON DALIUGE

Traditionally, radio astronomy data are acquired from a telescope and post-processed on personal computers. The next-generation large radio telescopes, represented by the SKA, will generate enormous amounts of data and require high-performance software running on supercomputers to process them properly (An 2019). Processing pipelines on large supercomputers must run in near real time and be managed by an execution framework (Banyer, Murphy &

Downloaded from https://academic.oup.com/mnras/article/526/2/1809/7286492 by University of Wisconsin Milwaukee user on 30 September 2024

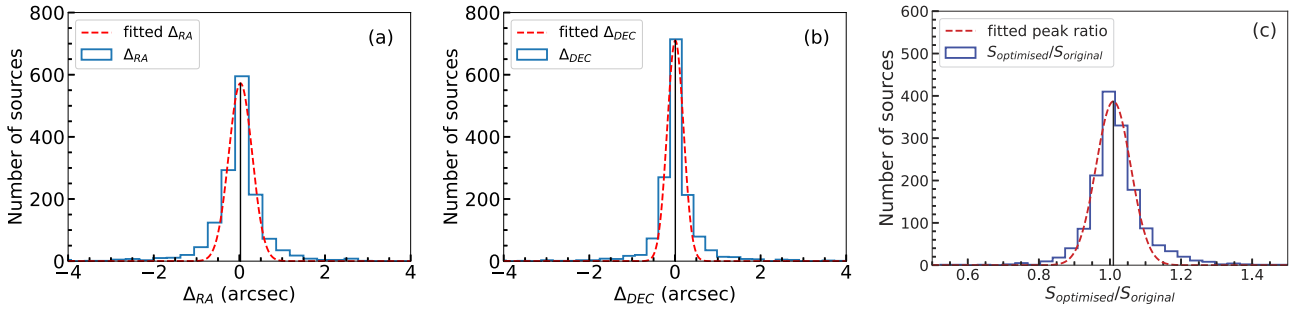

Figure 5. Comparison of the sources detected from the optimized and original pipelines: (a) difference of right ascension Δ_{RA}; (b) difference of declination $\Delta_{Dec.}$; (c) flux density ratio.

Table 3. Comparison of the matched candidates between optimized and original pipelines.

Methods	Source name	RA (deg)	Dec. (deg)	S_{peak_flux} (mJy beam^{-1})	S_{int_flux} (mJy)	$\tilde{\chi}^2_{lc}$	m (per cent)
Optimized pipeline	J005800.93−235449.08	14.503895	−23.913633	10.46 ± 0.034	10.45 ± 0.034	172.5	21.1
Original pipeline	J005800.93−235449.02	14.503886	−23.913618	10.50 ± 0.043	10.53 ± 0.044	126.5	20.9
Optimized pipeline	J005806.74−234744.51	14.528084	−23.795696	8.47 ± 0.033	8.42 ± 0.032	4.6	4.0
Original pipeline	J005806.74−234744.48	14.528072	−23.795689	8.44 ± 0.042	8.53 ± 0.042	3.6	3.8
Optimized pipeline	J005808.98−233453.97	14.537424	−23.581660	1.47 ± 0.028	1.50 ± 0.029	5.0	20.6
Original pipeline	J005808.97−233453.92	14.537392	−23.581645	1.44 ± 0.035	1.64 ± 0.040	3.6	14.0
Optimized pipeline	J005812.02−233735.64	14.550077	−23.626566	5.13 ± 0.029	4.96 ± 0.029	65.9	22.9
Original pipeline	J005812.01−233735.56	14.550052	−23.626545	5.03 ± 0.037	4.92 ± 0.036	51.9	21.0
Optimized pipeline	J010130.80−235205.62	15.378351	−23.868228	0.35 ± 0.035	0.25 ± 0.025	5.8	107.8
Original pipeline	J010131.02–235205.77	15.379248	−23.868269	0.36 ± 0.048	0.32 ± 0.043	6.7	81.5

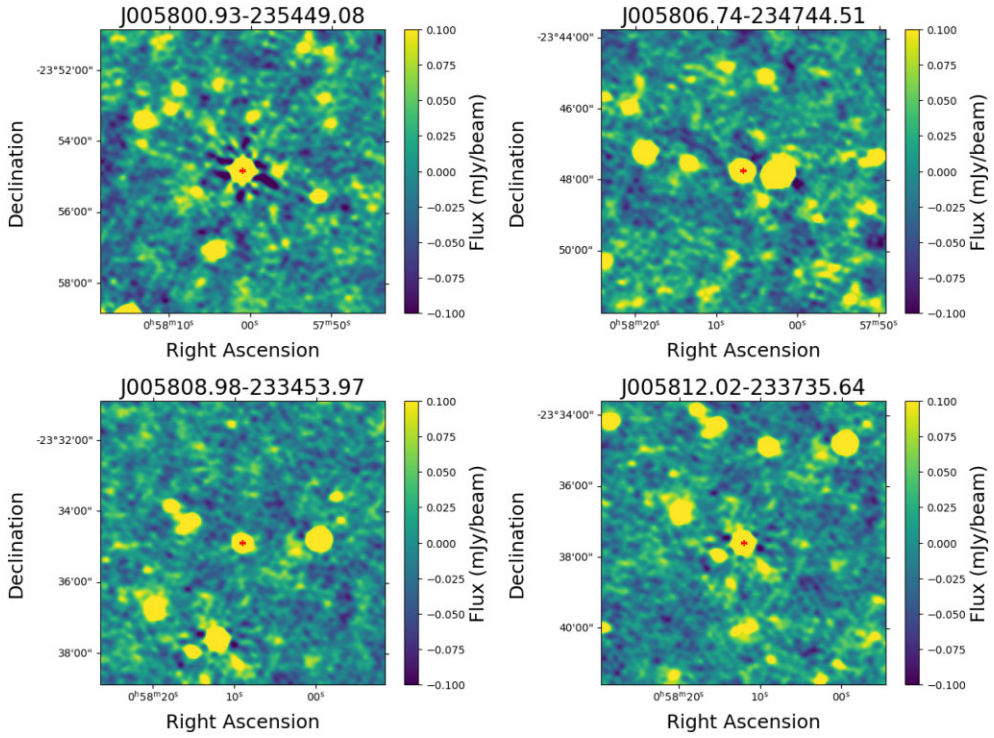

Figure 6. Cutout deep images of true variable sources detected by the optimized pipeline on SB9602_beam29. The red cross markers denote the peak positions of the variable sources.

Downloaded from https://academic.oup.com/mnras/article/526/2/1809/7286492 by University of Wisconsin Milwaukee user on 30 September 2024

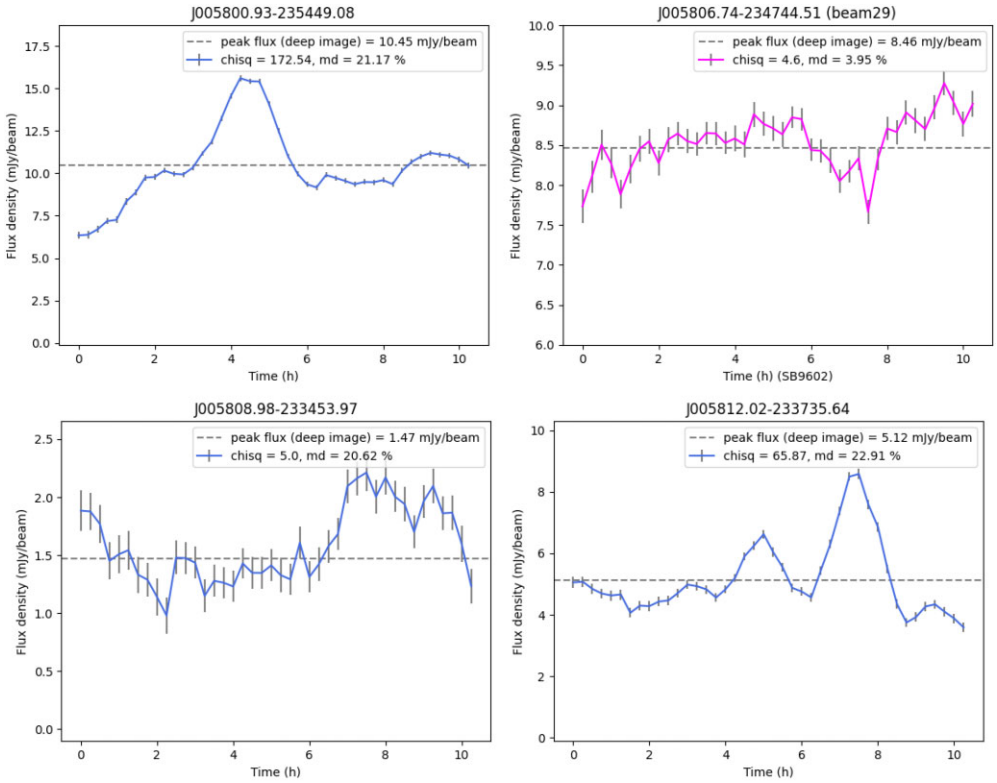

Figure 7. Light curves of true variable sources generated by the optimized pipeline on the SB9602_beam29 data.

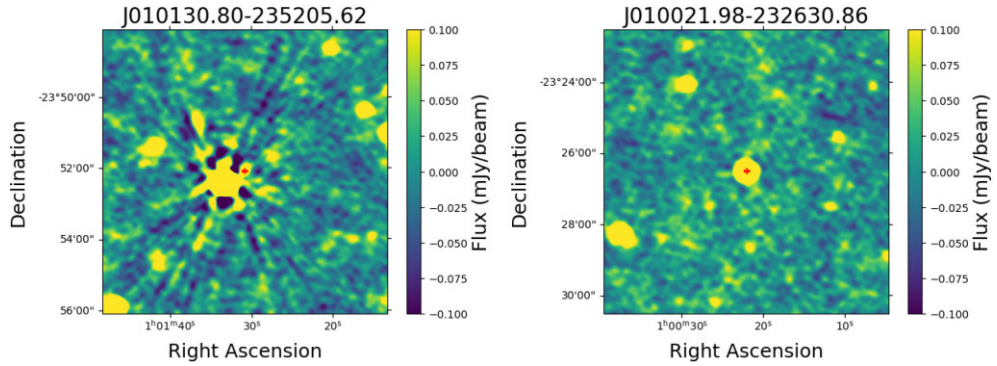

Figure 8. Two examples of false detections. The left-hand panel shows a candidate variable source (marked by a red cross) matched between the output images of two pipelines, but is a false detection due to the sidelobe contamination. The right-hand panel shows a candidate variable source detected by the optimized pipeline but not by the original pipeline. This is also a false detection because it is located further away from the centre of beam #29 and closer to beam #35. However, it did not appear in the catalogue of beam #35.

VAST Collaboration 2012). DALiuGE, a data-driven execution framework (Wu et al. 2017), was developed as a key component of the Data Flow Management System prototype for the SKA Science Data Processor (SDP; Broekema, van Nieuwpoort & Bal 2015).

DALiuGEhas demonstrated the power and scalability of near real-time data processing for large-scale surveys (Dodson et al. 2022; Ouyang, Lu & Lin 2022). The ASKAP data reduction software ASKAPSoft (Guzman et al. 2019) has been successfully run on DALiuGE (Guzman & Wicenec 2020). In order to implement a highly scalable automated VAST detection pipeline for the ongoing VAST full survey, we integrated the optimized pipeline on DALiuGE to accommodate massively parallel computing and large-scale data management.

The implementation of the VAST detection pipeline on DALiuGE involves the following three stages: AppDrop development, Logical Graph creation, and graph deployment. These are described below.

3.1 AppDrop development

Drop is the nomenclature for the basic components in DALiuGE, which represents the data or an application in pipelines, referred to as Data Drop (DataDrop) and Application Drop (AppDrop), respectively. DALiuGE is designed to provide a distributed data management platform and a scalable pipeline execution environment specifically to support exascale graph processing.

AppDrop development is to wrap the tasks of the pipeline into the form of Application Drops. Our development was based on the

Downloaded from https://academic.oup.com/mnras/article/526/2/1809/7286492 by University of Wisconsin Milwaukee user on 30 September 2024

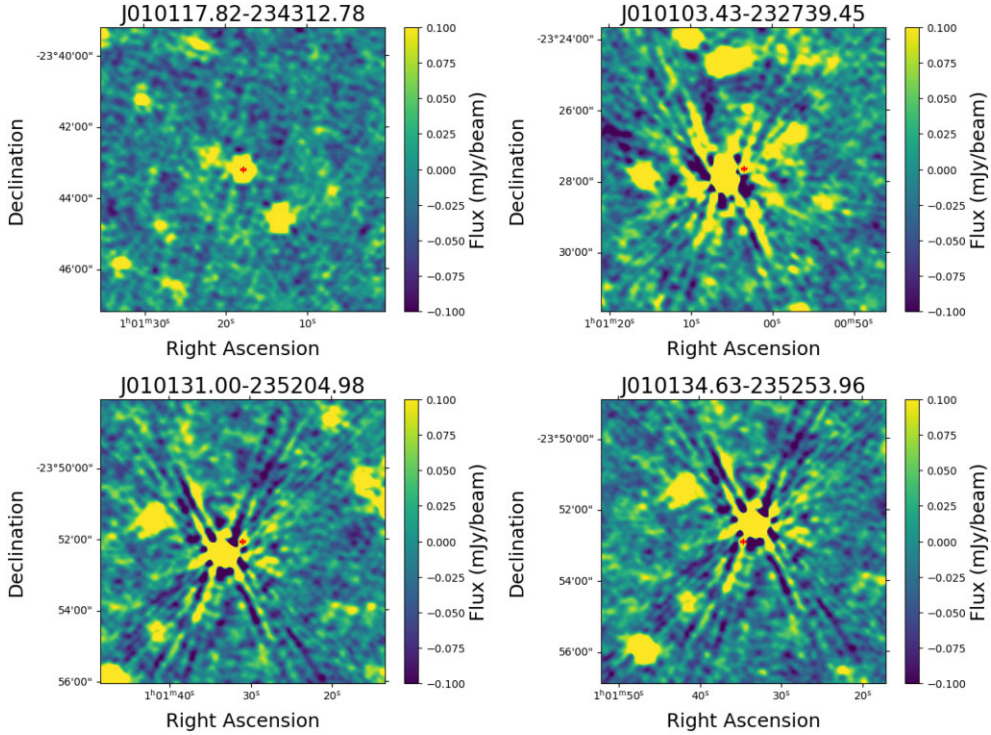

Figure 9. Candidate variable sources (marked by red crosses) detected by the original pipeline but not by the optimized pipeline. The top-left panel is regarded as a false detection as it is located further away from the centre of beam #29 and closer to beam #35. However, it did not appear in the results of beam #35. The other three are false detections appearing in sidelobes.

DALiuGE component project template[2], which provides a PYTHON AppDrop template as a starting point. We developed 10 AppDrops corresponding to the main tasks in the pipeline in Fig. 1.

3.2 Logical graph creation

We used the graph editor EAGLE[3] to connect the developed AppDrops into a scientific workflow called Logical Graph. Fig. 10 shows the Logical Graph created based on the workflow shown in Fig. 1. EAGLE uses JSON schemes to describe the components or Drops, with the description files called *palette*. For each AppDrop, the PYTHON script xml2palette.py from the DALiuGE repository[4] generates the EAGLE palette. In EAGLE, the output port of one *Drop* can connect to the input port of another *Drop* via an *edge* (arrows in the graph indicate connections between them). *Edge*s represent events triggered by *Drop*s, which in turn trigger processing in connected *Drop*s.

In Fig. 10, we use two *Scatter* components to parallelize the whole pipeline and steps like generating snapshot images from beam data segmentation (*Scatter1*) and estimating background and rms noise levels (*Scatter2*). *Scatter* handles parallel data processing. *Drop* group inside *Scatter* will be split to process different data segmentation. The input and output ports of each AppDrop connect to file or data *Drops*. These file *Drops* specify input and output data file paths and filenames for each AppDrop, automatically updated during graph execution. This embodies the data-driven concept.

[2]https://github.com/ICRAR/daliuge-component-template
[3]EAGLE is a visual paradigm-based web application that supports workflow integration for a variety of applications: https://github.com/ICRAR/EAGLE
[4]https://github.com/ICRAR/daliuge

3.3 Graph deployment

The Logical Graph in the previous section depicts the logical relationships of AppDrops. The first task in the graph deployment is for the DALiuGE translation engine to use the METIS algorithm (Karypis & Kumar 1998) to translate the logical graph into a physical graph template. The Physical Graph Template is the unfolding result of each complex component of the Logical Graph, and is a detailed description of each individual processing step of the final workflow. Taking Fig. 10 as an example, if the number of copies of *Scatter1* and *Scatter2* are both 1, the generated Physical Graph Template will have a total of 22 *Drops*. If the number of copies of *Scatter1* and *Scatter2* are 3 and 28, respectively, it will generate $3 \times (4 \times 28 + 18) = 390$ *Drops* in total.

Next, the Physical Graph Template is mapped on to specific computing resources (e.g. computer nodes) to generate the so-called Physical Graph. In the Physical Graph, each partition is mapped to a set of allocated resources, and each *Drop* is assigned a physical resource ID (e.g. IP address, hostname) using an optimal load-balancing approach.

Finally, the Physical Graph is submitted to the Drop Manager in DALiuGE for instantiating and deploying the *Drops* specified in the Physical Graph on its managed resources. The Drop Manager acquires the real-time resources of the compute clusters (i.e. CSRC-P in our experiments) and deploys the *Drops* to the computing resources via the graph partitioning algorithm (i.e. METIS) to maximize resource utilization and pipeline efficiency. After the manager sends a trigger event to initiate *Drops*, the Physical Graph can execute itself. During execution, the Drop manager monitors the execution progress by listening to *Drop* events.

DALiuGE is a powerful execution framework designed to handle large-scale astronomical data. Its execution involves a series of disc

Downloaded from https://academic.oup.com/mnras/article/526/2/1809/7286492 by University of Wisconsin Milwaukee user on 30 September 2024

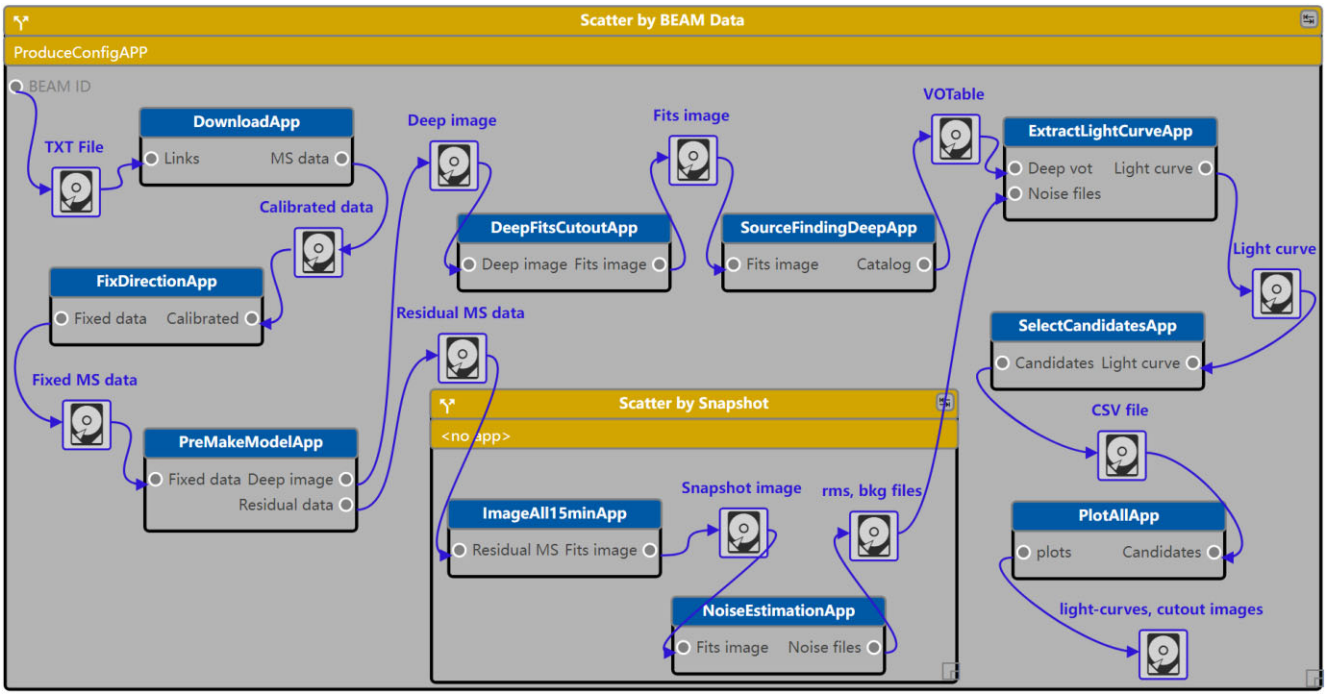

Figure 10. Logical graph of optimized pipeline running on the DALiuGE execution framework. The logical graph is a projection of the workflow shown in Fig. 1. The ten application drops are in sequence: DownloadApp (downloading calibrated ASKAP data from the CASDA), FixDirectionApp (fixing the observation phase centre), PreMakeModelApp (making a deep CLEAN model and subtracting the model from the visibility data), DeepFitsCutoutApp (cropping deep images into cutout images), SourceFindingDeepApp (source finding and detection), ImageAll15minApp (generating snapshot images), NoiseEstimationApp (generating residual images and estimating rms noise), ExtractLightCurveApp (extracting light curves), SelectCandidatesApp (identifying candidate variable sources), PlotAllApp (plotting light curves, cutout deep images and residual snapshot images containing identified variable candidates).

read/write operations. However, at the time of design, DALiuGE takes into account the bottlenecks that can be caused by disc input/output (I/O), and it does not necessarily require that each task writes data to and from the disc at all times (Wu et al. 2017). DALiuGE adopts a data-driven execution mechanism. Data-activated tasks are executed only when the input data is ready. This helps to mitigate potential performance issues associated with disc I/O operations.

In addition, DALiuGE uses a distributed design that can scale to handle larger data rates or computational problems. A typical application of DALiuGE is the completion of workflow simulations at the scale of the SKA phase I (SKA1). Wang et al. (2020b) successfully simulated and processed SKA1-scale data on the *Summit* supercomputer using DALiuGE, generating 2.6 Peta Byte (PB) of raw data in a workflow at a peak computing rate of 130 Peta Flops (PFlops). This is the first time that petabytes of radio astronomical data have been processed at a hundred PFlops in a workflow manner (Wang, Wicenec & An 2020a; Wang et al. 2020b). The results show that the DALiuGE workflow is capable of handling critical SKA science cases. In the current work, we successfully ran the VAST detection pipeline in parallel using DALiuGE.

3.4 Performance test and result analysis

The optimized VAST detection pipeline and the data obtained from the VAST pilot survey are used in this performance verification experiment. The observation ID is SB32039, and the phase centre of the observation is at RA = $19^{h}07^{m}58\overset{s}{.}6$, Dec. = $-64^{\circ}30'37''$, with a central frequency of 944 MHz, 1 MHz frequency resolution and 10 s integrations. The 7-h observation from 2021-09-10 UT 12:09 to UT 19:09.10 yields 36 beams of visibility data, ∼15 GB for each beam, resulting in a total data size of ∼540 GB. 16 of these beam data were used in our experiments, with the number of the beams chosen by the number of compute nodes used.

The experiments were conducted on the HPC in the China SKA Regional Centre, using a total of 16 Intel X86 compute nodes (specifications in Table 4).

To evaluate the DALiuGE performance, we parallelized the optimized pipeline using traditional parallel algorithms MPI and BASH Shell, comparing to DALiuGE. Each beam data is processed on a single compute node, and multiple beam data run on different nodes simultaneously using MPI, BASH Shell, and DALiuGE methods, respectively. We have run ten tests for each method on 1, 2, 4, 8, and 16 compute nodes, respectively.

Fig. 11 shows the total runtime. The runtime is the average of the 10 tests on the corresponding number of compute nodes. The error bar represents the standard deviation of the runtimes. It can be seen that the DALiuGE is the fastest for all data of various sizes. Moreover, the runtime of the DALiuGE-based execution does not show a significant increase with increasing data size (i.e. the number of beams). From 1 to 16 compute nodes, DALiuGE's runtime only varied 0.132 h, while MPI and BASH have a variation of 0.173 and 0.168 h, respectively. This indicates that DALiuGE has better stability. Future experiments using more nodes will further validate the scalability and stability of DALiuGE. Finally, DALiuGE automatically distributes triggered tasks based on heuristics like load balancing, reducing the need for complex distributed programming.

DALiuGE uses a graph structure to facilitate the workflow, and has the advantage over traditional parallel BASH or MPI routines in its ability to optimize the scheduling and execution of tasks based on

Downloaded from https://academic.oup.com/mnras/article/526/2/1809/7286492 by University of Wisconsin Milwaukee user on 30 September 2024

Table 4. Specifications of X86 compute nodes of the China SKA Regional Centre used for this study.

	Type	Quantity	Specification	Total
15 nodes	CPU	2	Intel Xeon Gold 5218 CPU@2.3GHz	32 Cores
	Memory	12	DDR4 RDIMM 2666MHz-64GB	768 GB
	Hard disc	2	600 GB-SAS 12Gb/s-15000rpm−2.5	1.2 TB
1 node	CPU	2	Intel Xeon Gold 6132 CPU@2.6GHz	28 Cores
	Memory	16	DDR4 RDIMM 2666MHz-64GB	1 TB
	Hard disc	2	SSD-2TB-NVMe	4 TB

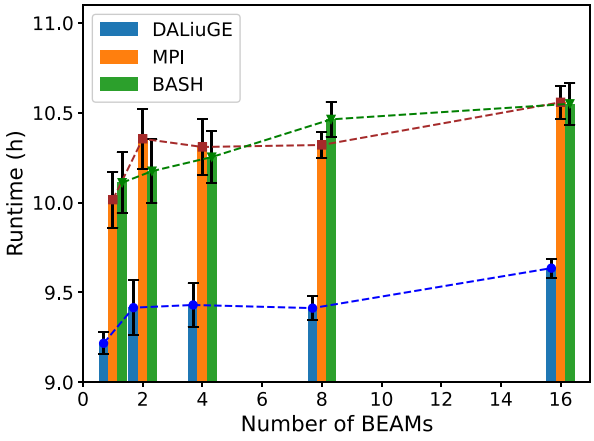

Figure 11. Comparison of the runtimes of DALiuGE, MPI, and BASH pipelines. The mean runtimes of the DALiuGE method are 9.42 h, 10.31 h for the MPI method, 10.31 h for the BASH method. The standard deviation of the runtime curve is 0.132 h for DALiuGE, 0.173 h for MPI, and 0.169 h for BASH.

their dependencies and data locations. In DALiuGE, *Drops* represent computations or data and *Edges* represent data dependencies between them (see also in Section 3.2). The *Drop* manager monitors the execution progress by collecting execution status via *Edges*. For SKA data processing, data movement is a critical factor in the overall execution time of a workflow. This intelligent task scheduling structure enables DALiuGE to know which computations depend on which other computations and schedule them to minimize data movement.

DALiuGE provides flexibility in data storage through its Data Drop design, supporting storage in memory, distributed object storage, and parallel file systems. DALiuGE's data-driven execution model also enables streaming pipelines to avoid unnecessary write/read operations. However, it should be noted that while these optimizations of data localization and movement can help mitigate the data I/O pressure, it has limitations when workflows generate huge intermediate data sets. Further analysis is needed on DALiuGE's performance in handling very large intermediate data or poor disc I/O performance to obtain quantitative evaluations. Addressing I/O bottlenecks in extreme environments requires continued integration of new storage and I/O technologies in execution framework development, along with performance tuning tailored to specific scientific applications.

4 SUMMARY AND PERSPECTIVE

This paper describes a method to improve and optimize the transient detection pipeline for the SKA slow transient search project (VAST).

We first replace the w-projection algorithm used in the original pipeline with the w-stacking CLEAN algorithm integrated into WSClean. This modification yielded two key improvements. First, the optimized pipeline exhibits superior parallelism and scalability compared to the original pipeline, resulting in a significant improvement in the multinode parallelism efficiency. This is most notable in the steps of deep sky model creation and snapshot image creation, with the overall execution efficiency of the entire pipeline being improved by a factor of about three. Secondly, the optimized pipeline generates residual images with lower noise levels, more uniform residual noise distributions and fewer artefacts compared to the original pipeline. This implies that the improved pipeline is not only faster but also produces higher fidelity results.

Furthermore, we integrated the optimized pipeline into the Data Activated *Liu* Graph Engine (DALiuGE) framework. This integration marks one of the first successful applications of the DALiuGE execution framework to SKA precursor data processing, enabling the imaging pipeline to adapt to large-scale, high-performance distributed computing tasks. This increases the operational efficiency of future SKA pipelines on HPC clusters and reduces data processing time. Implementing the DALiuGE pipeline consumes less time than traditional parallel methods (MPI and BASH) and is also more stable and scalable. The method presented here can be applied not only to the VAST project, but also to other ASKAP imaging surveys, such as the EMU and POSSUM. The aim of these improvements is to make the most efficient use of the massive amounts of data produced by radio astronomy surveys, thereby enabling more accurate and detailed observations of the Universe.

ACKNOWLEDGEMENTS

We acknowledge the reviewer for her/his constructive comments that help to improve this manuscript. This research is supported by the National SKA Program of China (2022SKA0130103) of the Ministry of Science and Technology of China and Youth Innovation Promotion Association fellowship of the Chinese Academy of Sciences (201664, 2021258). ZX and SG acknowledge support from the Laboratory of Pinghu. TA thanks Andreas Wicenec for his comments on DALiuGE. The experiments made use of the computing resource of the China SKA Regional Centre prototype, funded by the Ministry of Science and Technology of China (2018YFA0404603) and the Chinese Academy of Sciences. The Australian Square Kilometre Array Pathfinder is part of the Australia Telescope National Facility which is managed by 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

Downloaded from https://academic.oup.com/mnras/article/526/2/1809/7286492 by University of Wisconsin Milwaukee user on 30 September 2024

Government, with support from the Government of Western Australia and the Science and Industry Endowment Fund.

DATA AVAILABILITY

All of the ASKAP data used in this paper can be accessed through the CSIRO ASKAP Science Data Archive (CASDA)[5], the project codes and SBIDs are SB9602 in AS111 (ASKAP Pilot Survey for Gravitational Wave Counterparts) and SB32039 in AS107 (ASKAP Pilot Survey for VAST). The software packages used in this work can be obtained from public websites: CASA (use version 5.0.0 in this work) https://casa.nrao.edu/; WSClean (use version 2.9.0 in this work) https://gitlab.com/aroffringa/wsclean/; DALiuGE https://github.com/ICRAR/daliuge. Our optimized detection pipeline can be provided upon reasonable request. Deep images used in this study are available on GitLab at links: https://gitlab.com/csrc-p1/vast-fast-imaging/-/tree/main/deep_image/.

REFERENCES

Abbott B. P. et al., 2016, Phys. Rev. Lett., 116, 061102
Abbott B. P. et al., 2017, Phys. Rev. Lett., 119, 161101
An T., 2019, Sci. China Phys. Mech. Astron., 62, 989531
An T., Wu X.-P., Hong X., 2019, Nat. Astron., 3, 1030
An T., Wu X., Lao B., Guo S., Xu Z., Lv W., Zhang Y., Zhang Z., 2022, Sci. China Phys. Mech. Astron., 65, 129501
Anderson C. S. et al., 2021, PASA, 38, e020
Banyer J., Murphy T., VAST Collaboration 2012, in Ballester P., Egret D., Lorente N. P. F.eds, ASP Conf. Ser. Vol. 461, Astronomical Data Analysis Software and Systems XXI. Astron. Soc. Pac., San Francisco, p. 725
Bell M. E. et al., 2014, MNRAS, 438, 352
Bellm E. C. et al., 2019, PASP, 131, 018002
Bonaldi A. et al., 2021, MNRAS, 500, 3821
Bond I. A. et al., 2001, MNRAS, 327, 868
Broekema P. C., van Nieuwpoort R. V., Bal H. E., 2015, J. Instrum., 10, C07004
Cao Z. et al., 2021, Nature, 594, 33
Chapman J. M., Dempsey J., Miller D., Heywood I., Pritchard J., Sangster E., Whiting M., Dart M., 2017, in Lorente N. P. F., Shortridge K., Wayth R.eds, ASP Conf. Ser. Vol. 512, Astronomical Data Analysis Software and Systems XXV. Astron. Soc. Pac., San Francisco, p. 73
Cornwell T. J., 2008, IEEE J. Sel. Top. Signal Process., 2, 793
Cornwell T. J., Golap K., Bhatnagar S., 2005, in Shopbell P., Britton M., Ebert R.eds, ASP Conf. Ser. Vol. 347, Astronomical Data Analysis Software and Systems XIV. Astron. Soc. Pac., San Francisco, p. 86
Dewdney P. E., Hall P. J., Schilizzi R. T., Lazio T. J. L. W., 2009, IEEE Proc., 97, 1482
Dodson R. et al., 2022, AJ, 163, 59
Duchesne S. W. et al., 2023, PASA, 40, e034
Fender R. et al., 2016, Proc. Sci., MeerKAT Science: On the Pathway to the SKA. SISSA, Trieste, PoS#13
Gao H., Li Z., Zhang B., 2014, ApJ, 788, 189
Guzman J., Wicenec A., 2020, in Pizzo R., Deul E. R., Mol J. D., de Plaa J., Verkouter H.eds, ASP Conf. Ser. Vol. 527, Astronomical Data Analysis Software and Systems XXIX. Astron. Soc. Pac., San Francisco, p. 531
Guzman J. et al., 2019, Astrophysics Source Code Library, record ascl:1912.003
Hale C. L. et al., 2021, PASA, 38, e058
Hancock P. J., Trott C. M., Hurley-Walker N., 2018, PASA, 35, e011
Hotan A. W. et al., 2021, PASA, 38, e009
Hurley-Walker N. et al., 2022, Nature, 601, 526
IceCube Collaboration, 2018a, Science, 361, 147
IceCube Collaboration, 2018b, Science, 361, eaat1378
Ivezić Ž. et al., 2019, ApJ, 873, 111
Karypis G., Kumar V., 1998, J. Parallel Distrib. Comput., 48, 96
Lacy M. et al., 2020, PASP, 132, 035001
Lao B., An T., Wang A., Xu Z., Guo S., Lv W., Wu X., Zhang Y., 2021, Sci. Bull., 66, 2145
Law C. J. et al., 2015, ApJ, 807, 16
McMullin J. P., Waters B., Schiebel D., Young W., Golap K., 2007, in Shaw R. A., Hill F., Bell D. J.eds, ASP Conf. Ser. Vol. 376, Astronomical Data Analysis Software and Systems XVI. Astron. Soc. Pac., San Francisco, p. 127
Murphy T. et al., 2013, PASA, 30, e006
Murphy T. et al., 2021, PASA, 38, e054
Norris R. P. et al., 2021, PASA, 38, e046
Offringa A. R., Smirnov O., 2017, MNRAS, 471, 301
Offringa A. R. et al., 2014, MNRAS, 444, 606
Ouyang N., Lu S., Lin L., 2022, in 2022 IEEE 10th International Conference on Computer Science and Network Technology (ICCSNT). Dalian, China, p. 140,
Pintaldi S., Stewart A., O'Brien A., Kaplan D., Murphy T., 2022, in Ruiz J. E., Pierfedereci F., Teuben P.eds, ASP Conf. Ser. Vol. 532, Astronomical Data Analysis Software and Systems XXX. Astron. Soc. Pac., San Francisco, p. 333
Taylor M. B., 2005, in Shopbell P., Britton M., Ebert R.eds, ASP Conf. Ser. Vol. 347, Astronomical Data Analysis Software and Systems XIV. Astron. Soc. Pac., San Francisco, p. 29
Wang R., Wicenec A., An T., 2020a, Sci. Bull., 65, 337
Wang R. et al., 2020b, SC20: International Conference for High Performance Computing, Networking, Storage and Analysis. IEEE, Atlanta, United States
Wang Y., Tuntsov A., Murphy T., Lenc E., Walker M., Bannister K., Kaplan D. L., Mahony E. K., 2021, MNRAS, 502, 3294
Wang Y. et al., 2023, MNRAS, 523, 5661
Wijers R. A. M. J., Bloom J. S., Bagla J. S., Natarajan P., 1998, MNRAS, 294, L13
Williams P., Allers K., Biller B., Vos J., 2019, Res. Notes Am. Astron. Soc., 3, 110
Wu C. et al., 2017, Astron. Comput., 20, 1
van Diepen G. N. J., 2015, Astron. Comput., 12, 174

This paper has been typeset from a TEX/LATEX file prepared by the author.

[5] https://data.csiro.au/dap/public/casda/casdaSearch.zul

© 2023 The Author(s)
Published by Oxford University Press on behalf of Royal Astronomical Society

Downloaded from https://academic.oup.com/mnras/article/526/2/1809/7286492 by University of Wisconsin Milwaukee user on 30 September 2024