An Initial Framework for Prototyping Radio-Inteferometric Imaging Pipelines

Sunrise Wang[1[0000−0002−5038−9531]], Nicolas Gac[1[0000−0001−6981−0368]], Hugo Miomandre[2[0009−0005−4832−3292]], Jean-Francois Nezan[2[0000−0002−0609−4592]], Karol Desnos[2[0000−0003−1527−9668]], and Francois Orieux[1[0000−0001−5638−3416]]

[1] Université Paris-Saclay, CNRS, CentraleSupélec, Laboratoire des signaux et systèmes, 91190, Gif-sur-Yvette, France sunrise.wang@centralesupelec.fr {nicolas.gac,francois.orieux}@universite-paris-saclay.fr

[2] Univ Rennes, INSA Rennes, CNRS, IETR – UMR 6164, F-35000 Rennes, France {hugo.miomandre,jean-francois.nezan,karol.desnos}@insa-rennes.fr

Abstract. Although large radio-telescope arrays allow us to observe the celestial sphere with an unprecedented level of detail and sensitivity, additional antennas drastically increases the cost of processing and storing their data, complicating the design of computing hardware. Our overall goal is to provide a system, which we term SimSDP, to aid in their design. It will achieve this by providing resource usage estimations for some given imaging pipeline and hardware architecture, allowing for more informed decisions when building the production systems. We lay the groundworks in this paper by presenting and validating an initial system that implements three different imaging pipelines. We find that in most cases, our system is able to accurately estimate the scaling across both algorithmic parameters as well as parallelization when compared to measured data, with errors roughly in the 1-5% range, demonstrating its ability to inform design decisions.

Keywords: Algorithm Design Space Exploration · SKAO · PREESM · Radio-Interferometry · Resource Estimation

1 Introduction

Radio-telescopes capture information of our skies within the radio spectrum, allowing for the study of a host of otherwise invisible natural phenomena, with some examples including unionized gas clouds, and low-frequency synchotron radiation. Although many initial radio-telescopes were single-dish instruments, modern advancements in the field of aperture synthesis allows us to instead use antenna arrays with radio-interferometry. This is advantageous as it allows for much larger apertures and higher sensitivities, allowing us to observe smaller and fainter objects.

The currently under construction Square Kilometer Array (SKA) is an example of such a telescope, and on completion, will be the largest antenna array in the world. An issue with the SKA, as well as other large antenna arrays, is their

computational cost, as the raw data obtained from the antennas is processed digitally. Thus, the more antennas, the larger the amount of data to be processed, with estimates for the SKA being around 0.4TB/s, leading to roughly 34.5 PBs per day. Storing this data for any significant period of time is prohibitively expensive, leading to strict time restrictions during processing. This, coupled with the fact that the type of science performed dictates the algorithmic pipeline, makes hardware and software design particularly challenging.

To aid in the above, a system that provides resource usage estimations for various radio-interferometric algorithmic pipelines and hardware architectures is desired. Unfortunately, as far as we are aware, none currently exists. The closest to this is DALiuGE [1], a graph-based execution framework targeted at radio-astronomy, and DASK [2], which has been used in radio-interferometry systems such as RASCIL [3]. However, from our understanding, these are primarily execution frameworks and do not provide easy resource usage estimations, particularly at scale.

It is our aim to introduce such a system, which will be termed SimSDP as it is targeted at the design of the SKA supercomputer, the Science Data Processor (SDP). This system will consist of two main parts. The first is responsible for both the algorithmic descriptions, as well estimating the performance for hardware at a local level, and the second aims to generalize the performance estimations across different hardware architectures.

This paper lays the groundwork for the former, by presenting a generic framework for radio-inteferometric algorithms, and providing three implementations of imaging pipelines within. We evaluate our framework's ability to aid in the design of the SDP by estimating the resource usage of these different pipelines across a variety of parameters and two levels of parallelization, and compare these against measured data to see if we can draw similar conclusions.

The main contributions of this paper are:

- A generic framework describing radio-interferometric algorithms, together with the full data-flow description and implementation of three different imaging pipelines,
- An evaluation of our framework's ability to aid in the design of the SDP, by comparing both estimated resource usages against measured across a variety of parameters and two parallelization levels,

The remainder of this paper is structured as follows: Section 2 provides a brief introduction to radio-interferometry as well as the generic pipeline, Section 3 discusses our implementation of this, and provides details on our implemented algorithms and how we perform our estimations, Section 4 discusses our results, and Section 5 concludes our work.

2 Radio-Interferometric Imaging

Radio-interferometers measure the sky using antenna arrays. Samples are produced by pairs of antennas in the array, termed baselines. Each sample, termed a

visibility, is the instrumental response for some given time duration and electromagnetic frequency for a specific baseline. Visibilities can be defined with the Radio-Interferometric Measurement Equation (RIME) [4]:

$$V(u,v) = G_{uv} \int D_{uv}(l,m) \frac{1}{\sqrt{1-l^2-m^2}} I(l,m) e^{-2\pi i(ul+vm+w(\sqrt{1-l^2-m^2}-1))} \mathrm{d}l\mathrm{d}m \quad (1)$$

where G denotes the direction independent effects, such as antenna gain, D denotes the direction dependent effects, such as Faraday rotation caused by the Earth's ionosphere, (u,v) refers to the difference between antenna positions in the frame of the Earth's rotation, (l,m) refers to spatial coordinates on the celestial sphere, and I is the true sky.

If the D and $e^{w(\sqrt{1-l^2-m^2}-1)}$ terms are ignored, Equation 1 simplifies to

$$V(u,v) = G_{uv} \int I(l,m) e^{-2\pi i(ul+vm)} \mathrm{d}l\mathrm{d}m \quad (2)$$

a 2-dimensional Fourier transform, allowing us to retrieve an image of the true sky through an inverse Fourier transform.

The image produced by the inversion of Equation 2, termed the dirty image, contains artefacts caused both by the partial sampling of the Fourier domain, as antenna arrays are inherently sparse, as well as by the omission of the w and D terms. The former results in a convolution between the Fourier transform of the sampling pattern, ie. the Point Spread Function (PSF), and the real sky, while the latter results in non-isoplanatic image-plane distortions. Radio-interferometric imaging aims primarily to correct these in order to produce an image use-able for scientific purposes.

The Radio-Inteferometric pipeline [5] employs a nested loop structure (Figure 1). The outer ie. major loop Δ calculates the difference $\delta\hat{i}_n$ between some estimated sky image $\hat{i}_n$ and the measured values v, with $\hat{i}_0$ typically being a blank image.

Although there are many possible approaches to Δ, the most common general method currently involves transforming $\hat{i}_n$ to the same domain as v using a degridding operator, which typically includes a Fourier transform and an extrapolation operator, performing the subtraction to obtain the difference in visibility space, and then using a gridding operator which includes an inverse Fourier transform together with an interpolation operator to obtain $\delta\hat{i}_n$.

In addition to computing the difference, the major loop is also responsible for correcting for the w and D terms. This can be achieved either in tandem with the gridding convolution stage $G^\dagger$ [6–8] or through various discretization methods [9–11]. There has also been work that employs a hybrid of the two strategies [12, 13].

Finally, the major loop also allows for calibration of the direction independent effects G, which we don't show in Figure 1 as it is not the focus of our work.

The inner ie. minor loop Ψ is responsible for removing the convolution artefacts caused by the partial sampling of the Fourier domain from $\delta\hat{i}_n$, and then combining this information with $\hat{i}_n$ to produce an estimate $\hat{i}_{n+1}$ for the next major cycle iteration. This is achieved by employing a deconvolution algorithm, with

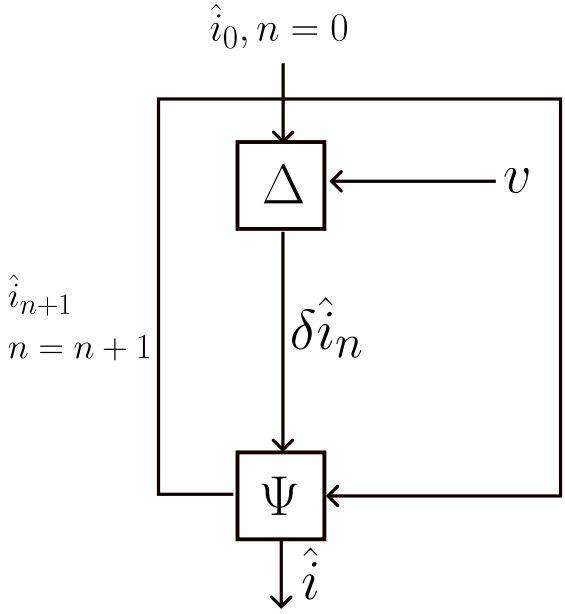

Fig. 1. The general framework for producing the final sky estimate $\hat{i}$ is iterative, where $\hat{i}_n$ is updated while taking into account the difference $\delta\hat{i}_n$ between it and the measurements v. The degridding-gridding step (Δ) is responsible for computing this, whereas the deconvolution step Ψ is responsible for generating the next estimate $\hat{i}_{n+1}$.

some popular examples being CLEAN [14] and its variants [15–17], and convex optimization methods that regularize based on entropy [18] or sparsity [19–21].

As the deconvolution methods can be partial or contain errors in their estimation, the new estimate $\hat{i}_{n+1}$ is passed back to the major loop. This is done until $\delta\hat{i}_n$ is predominantly noise.

3 Implementation

We implement the generic structure defined in Figure 1, as well as three different imaging algorithms using PREESM [22]. We find PREESM to be particularly well suited to our needs, as algorithms are defined in data-flow diagrams, which allows for easy definition of additional imaging algorithms. It also allows for automated parallelism, which we use for our different levels of parallelization. Finally, PREESM also allows for easy estimation of resource consumption when provided with the relevant data for each actor.

The generic top-level data-flow diagram provides a common interface for algorithms defined in Δ and Ψ ie. degridding-gridding and deconvolution. This enables us to encompass a large number of different imaging pipelines, as most work in the domain tend to focus on either.

The rest of this section will look to provide a high-level overview on the algorithms we implemented for both Δ and Ψ, as well as how parallelization is

performed. It also discusses how we perform our resource usage estimations. We provide our system and data in our on-line repository[3].

3.1 Algorithms for Δ and Ψ

We implement three different degridding-gridding algorithms (Δ) in our pipeline. The first is the Direct Fourier Transform (DFT) degridder, which transforms the image directly to the visibilities using:

$$V(u,v) = \sum_{l=0,m=0}^{L,M} I(l,m)e^{-2\pi i(ul+vm)} \tag{3}$$

As it is unnecessary to perform the equation for areas where I is zero, the overall complexity for this method is $\mathcal{O}(n_v n_s)$, where n_v is the total number of visibilities, and n_s is the number of sources.

Following this is the Fast-Fourier Transform (FFT) degridder, which first performs a Fast Fourier Transform on the image to obtain the gridded visibilities, which are then subsequently convolved onto the continuous visibility positions. This algorithm has a complexity of $\mathcal{O}(n_g^2 \log_2 n_g + n_v n_{dgk})$, where n_g is the support size of the grid, and n_{dgk} is the support size of the de-gridding kernel.

Both the DFT and FFT degridders also perform a gridding operation after the subtraction of visibilities, which has a complexity of $\mathcal{O}(n_g^2 \log_2 n_g + n_v n_{gk})$ where n_{gk} is the size of the gridding kernel.

The last implemented degridding-gridding algorithm is the more recent Grid to Grid (G2G) method by Monnier et al. [23]. This method performs degridding and gridding in a single step, and performs the subtraction in image space rather than visibility. The total complexity of this algorithm is $\mathcal{O}(2n_g^2 \log_2 n_g + 2n_v(n_{gk} + n_{dgk}))$, the same as the FFT degridder. However, this algorithm is more performant, as it employs a visibility simplification stage which diminishes the size of n_v. Furthermore, the subtraction is done in image space, which may be faster as the number of visibilities often dwarfs the number of pixels in the image.

We choose to focus on degridding-gridding (Δ) algorithms for this work, thus only implement a single deconvolution algorithm (Ψ), that being Högbom CLEAN. [14], which we selected for its simplicity. This algorithm iteratively finds the brightest source in the image, and subtracts the PSF from its position, terminating after a preset number of minor cycles n_m. The complexity of this algorithm is $\mathcal{O}(n_g^2 + n_m n_p)$, where n_p is the support of the PSF. As this algorithm finds a single point source per iteration, $n_s = n_m$, which is pertinent for the DFT degridder.

[3] https://gitlab.com/igorawratu/an-initial-framework-for-prototyping-radio-inteferometric-imaging-pipelines

3.2 Parallelization

To parallelize Δ, we design the data-flow diagram to divide the visibilities based on some specified number of threads, grid or degrid each subset of visibilities independently, and then merge each thread's results. This manner of parellelization is natural for Δ as it requires looping through the visibilities. We don't parallelize Ψ as Högbom CLEAN has cross iteration data dependencies, drastically complicating process.

We create two different levels of parallelization with 1 and 4 cores. We set the data division parameter in our data-flow diagrams to not divide the visibilities in the former case, and divide them into 4 disjoint subsets in the latter.

3.3 Estimating Resource Utilization

We perform computation time and memory estimations for our system. For the former, PREESM requires time information for each actor, either in the form of an exact value, or some function dependent on the input data. We opt for the latter as it affords us the ability to predict how the algorithms scale.

We perform benchmarking on optimized versions of the code for each actor, while varying the number of visibilities, the image size, and the number of minor cycle iterations. We then fit polynomials of the appropriate degree to these, which we then transfer to PREESM. We average 30 samples for each benchmark data point.

We perform the benchmarking for actors that contain simple and near-optimal implementations directly in PREESM. For more complex actors, such as the G2G and FFT degridding-gridding algorithms, we profile the code of Monnier et al. [23].

The dataset used to generate these benchmarks is obtained from the SKAO simulated database repository. We provide full information on this in Section 4.1. In order to vary the number of visibilities, we either truncate or duplicate them. We opt for this over random generation because the antenna layout may have an effect on the performance of some algorithms, such as G2G.

The machine used to obtain the benchmarks is a Dell Precision 3551, with an Intel Core i7 10875H 10th generation processor and 64GB of RAM. Re-profiling and fitting is required for other hardware as our fitted models become invalid.

Unlike the computational time estimations, memory estimations in PREESM are much less involved, with PREESM simply reporting the maximum allocated inter-actor memory. It does not estimate intra-actor memory as it does not perform any code analysis, and primarily serves as a lower-bound estimation.

4 Results and Discussion

This section provides details on the simulated dataset that we use for our experiments, our framework validation, as well as our resource estimation results.

4.1 Simulated Database

The simulated dataset used for our validation and experiments is the same as the one used for our benchmarking, and is the small database from the SKA simulated dataset repository [4]. It is simulated to have a field of view of 1 degree, with 33 point sources within. The array has a total of 512 antennas, leading to 130816 baselines. The visibilities were sampled from each baseline once every 30 seconds, for a total of 15 minutes, creating 3924480 total visibilities.

This dataset is single polarization single wavelength, which is well suited to the current limitations of our system, as it is currently unable to handle different polarizations, extended sources, and multiple frequency channels.

4.2 Pipeline Validation

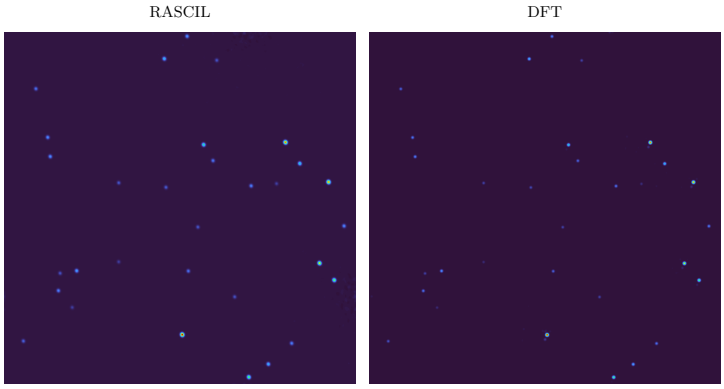

Fig. 2. Final deconvolved images of RASCIL and our DFT degridder-gridder. We can see that the sources appear in the same positions with similar energy ratios, showing that our pipeline functions as intended. Our other images are similar and available on our repository.

We validate our pipeline by performing a visual inspection of its reconstructed images to the ones produced from the well known imaging framework RASCIL [3]. Figure 2 shows the images produced both by RASCIL, and our DFT degridder. The other algorithms produce very similar images, and are available on our repository.

One can see that the general structure of the produced image is the same, with the point sources and their energy ratios being similar. The differences are primarily due to minor algorithmic discrepancies, such as parameters in the deconvolution algorithm.

[4] https://gitlab.com/ska-telescope/sim/sim-datasets

4.3 Computing Time Estimations

We evaluate our computation time estimations by comparing them to the measured times of our pipelines obtained using the GNU time tool. We plot these against different numbers of grid cells, numbers of visibilities, and the numbers of minor cycles per major iteration.

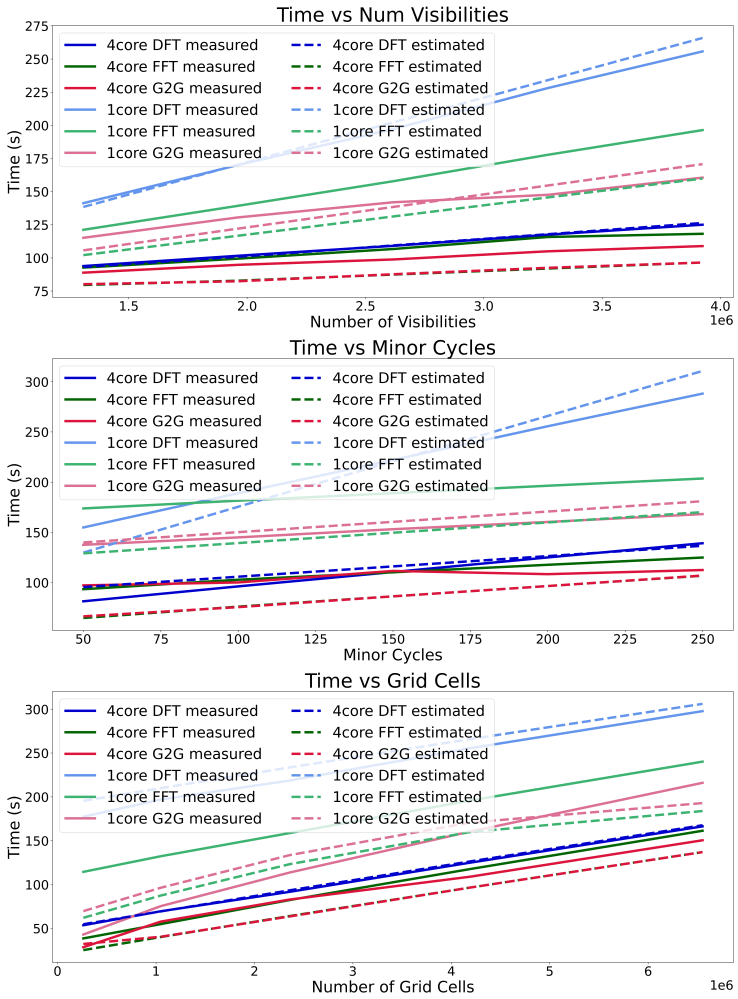

Fig. 3. Computational times of both our estimations (dashed lines) and measurements (solid lines) for our implemented methods across both levels of parallelization.

Figure 3 shows our results, where the dashed line denotes estimated times, the solid line denotes measured, the lighter colours denote the single core setup, and the darker denote the quad core.

We found our estimations to accurately predict the increase in time with increasing algorithm parameters, with most cases having an average error within the 1-5% range, allowing us to draw similar scaling conclusions to the measured data. For example, we can see that the DFT algorithm scales much poorer compared to the others with regards to the number of sources in the image, as well as the number of visibilities.

We also found that our framework allows us to draw valid conclusions with regards to scaling in terms of parallelization, with it accurately predicting in most cases the roughly $1.5 - 3\times$ speedup obtained when parallelizing across 4 cores.

Despite the positives, we found various areas in which the accuracy of our estimations was lacking. This can be seen for the FFT and G2G algorithms when varying the number of grid cells, and DFT in the cases where the minor cycles are varied. There are three main reasons for these poor estimates:

- With the exception of the DFT, the code benchmarked was much more optimized than the code measured;
- Our fitting functions introduce additional error;
- PREESM does not perform static analysis.

Although the former two points can be easily resolved by taking more accurate and a larger number of benchmark samples, the latter is a drawback to our current framework, which only performs static analysis. This is particularly evident in the results for G2G, which performs a visibility simplification step to dynamically reduce the number of visibilities needing to be processed.

4.4 Memory Usage Estimations

We evaluate the memory usage estimations by comparing them against the measured maximum resident set size obtained by the GNU time tool in verbose mode. We plot the memory usage against both the number of grid cells and the number of visibilities for both levels of parallelization. Unlike the computational time experiments, we do not vary the number of minor cycles as it does not have a major effect on memory usage.

Figure 4 shows the estimated and measured memory usage in our system. Much like the computation time estimations, our framework predicts well the increase when scaling the amount of data, with an error of 1-5% for all test cases. It also predicts the $1.1 - 1.5\times$ increase in memory usage when increasing parallelization.

These results are not surprising, as PREESM is responsible for allocating the inter-actor memory. However, it is unexpected that the measured memory usage is lower than the estimated. As mentioned in Section 3.3, the estimate should provide a lower-bound, which is in conflict with what we observed. We are still investigating possible explanations for this discrepancy, and will leave a more detailed analysis for future work.

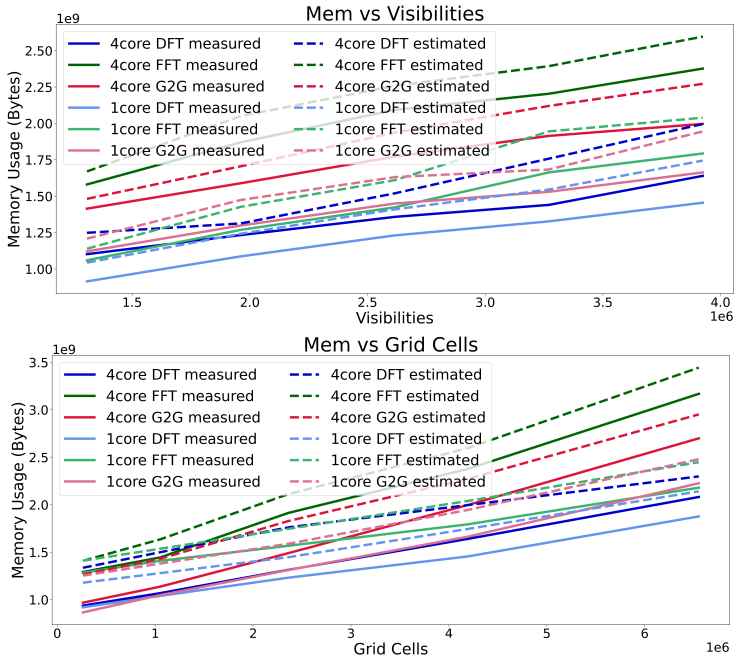

Fig. 4. Memory estimations (dashed lines) and measurements (solid lines) for the various algorithms implemented in PREESM plotted against both number of visibilities, and number of grid cells.

5 Conclusions and Future Work

In this paper, we lay the groundworks for SimSDP, a system aimed at aiding in the design of supercomputers processing data from large radio-telescope arrays by providing resource usage estimations at various scales and hardware architectures. We achieve this by performing a study on a preliminary system aimed at such, which includes three different algorithmic pipelines, and two levels of parallelization. We found that the estimations obtained from our framework allows us to draw similar scaling conclusions to the measured results. This gives impetus in continuing the design and implementation of SimSDP.

5.1 Drawbacks

One major drawback of our system is that it currently only performs static analysis, which may lead to it providing poor estimates in cases where algorithms perform dynamic reductions to the data being processed. Also in line with this, there are additional accuracy improvements that can be made to our estimations through more and better benchmarking data. Finally, our system currently does not handle many types of datasets, as it doesn't currently support objects containing extended sources (eg. nebulae), different polarizations, and multiple wavelengths.

5.2 Future Work

Much work still needs to be done for the realization of SimSDP. This includes: extending the current system to support different hardware architectures; implementing more algorithms for Δ and Ψ, which will allow for our system to support more types of datasets; introduce dynamic analysis by employing SPIDER [24]; estimating energy consumption; and comparing our system's estimations to measurements obtained from production systems [3, 11, 25].

Acknowledgements This work was supported by DARK-ERA (ANR-20-CE46-0001-01)

References

[1]C. Wu et al., "DALiuGE: A graph execution framework for harnessing the astronomical data deluge", en, Astronomy and Computing **20**, 1–15 (2017) 10.1016/j.ascom.2017.03.007.

[2]M. Rocklin, "Dask: Parallel Computation with Blocked algorithms and Task Scheduling", in (2015), pp. 126–132, 10.25080/Majora-7b98e3ed-013.

[3]T. J. Cornwell et al., *Radio astronomy simulation, calibration and imaging library*, 2020.

[4]O. M. Smirnov, "Revisiting the radio interferometer measurement equation: I. A full-sky Jones formalism", Astronomy & Astrophysics **527**, A106 (2011) 10.1051/0004-6361/201016082.

[5]B. Clark, "An efficient implementation of the algorithm 'CLEAN'", Astronomy and Astrophysics, vol. 89, no. 3, Sept. 1980, p. 377, 378. **89**, 377 (1980).

[6]T. J. Cornwell et al., "The Noncoplanar Baselines Effect in Radio Interferometry: The W-Projection Algorithm", IEEE Journal of Selected Topics in Signal Processing **2**, 647–657 (2008) 10.1109/JSTSP.2008.2005290.

[7]S. Bhatnagar et al., "Correcting direction-dependent gains in the deconvolution of radio interferometric images", Astronomy & Astrophysics **487**, 419–429 (2008) 10.1051/0004-6361:20079284.

[8]S. Van Der Tol et al., "Image Domain Gridding: a fast method for convolutional resampling of visibilities", Astronomy & Astrophysics **616**, A27 (2018) 10.1051/0004-6361/201832858.

[9]S. M. Ord et al., "Interferometric Imaging with the 32 Element Murchison Wide-Field Array", en, Publications of the Astronomical Society of the Pacific **122**, 1353–1366 (2010) 10.1086/657160.

[10]T. J. Cornwell and R. A. Perley, "Radio-interferometric imaging of very large fields-the problem of non-coplanar arrays", Astronomy and Astrophysics (ISSN 0004-6361), vol. 261, no. 1, p. 353-364. **261**, 353–364 (1992).

[11]A. R. Offringa et al., "Wsclean: an implementation of a fast, generic wide-field imager for radio astronomy", en, Monthly Notices of the Royal Astronomical Society **444**, 606–619 (2014) 10.1093/mnras/stu1368.

[12]T. J. Cornwell et al., "Wide field imaging for the square kilometre array", in, edited by P. J. Bones et al. (Oct. 2012), p. 85000L, 10.1117/12.929336.

[13]H. Ye et al., "High accuracy wide-field imaging method in radio interferometry", Monthly Notices of the Royal Astronomical Society **510**, 4110–4125 (2022) 10.1093/mnras/stab3548.
[14]J. A. Högbom, "Aperture synthesis with a non-regular distribution of interferometer baselines", Astronomy and Astrophysics Supplement, Vol. 15, p. 417 **15**, 417 (1974).
[15]F. R. Schwab, "Relaxing the isoplanatism assumption in self-calibration; applications to low-frequency radio interferometry", The Astronomical Journal **89**, 1076 (1984) 10.1086/113605.
[16]T. J. Cornwell, "Multiscale CLEAN Deconvolution of Radio Synthesis Images", IEEE Journal of Selected Topics in Signal Processing **2**, 793–801 (2008) 10.1109/JSTSP.2008.2006388.
[17]U. Rau and T. J. Cornwell, "A multi-scale multi-frequency deconvolution algorithm for synthesis imaging in radio interferometry", Astronomy & Astrophysics **532**, A71 (2011) 10.1051/0004-6361/201117104.
[18]T. J. Cornwell and K. F. Evans, "A simple maximum entropy deconvolution algorithm", Astronomy and Astrophysics (ISSN 0004-6361), vol. 143, no. 1, Feb. 1985, p. 77-83. **143**, 77–83 (1985).
[19]Y. Wiaux et al., "Compressed sensing imaging techniques for radio interferometry", en, Monthly Notices of the Royal Astronomical Society **395**, 1733–1742 (2009) 10.1111/j.1365-2966.2009.14665.x.
[20]R. E. Carrillo et al., "PURIFY: a new approach to radio-interferometric imaging", en, Monthly Notices of the Royal Astronomical Society **439**, 3591–3604 (2014) 10.1093/mnras/stu202.
[21]R. Ammanouil et al., "A parallel and automatically tuned algorithm for multispectral image deconvolution", Monthly Notices of the Royal Astronomical Society **490**, 37–49 (2019) 10.1093/mnras/stz2193.
[22]M. Pelcat et al., "Preesm: A dataflow-based rapid prototyping framework for simplifying multicore DSP programming", in 2014 6th European Embedded Design in Education and Research Conference (EDERC) (Sept. 2014), pp. 36–40, 10.1109/EDERC.2014.6924354.
[23]N. Monnier et al., "Fast Sky to Sky Interpolation for Radio Interferometric Imaging", in 2022 IEEE International Conference on Image Processing (ICIP) (Oct. 2022), pp. 1571–1575, 10.1109/ICIP46576.2022.9897317.
[24]H. Miomandre et al., *Demonstrating the SPIDER Runtime for Reconfigurable Dataflow Graphs Execution onto a DMA-based Manycore Processor*, IEEE International Workshop on Signal Processing Systems, Poster, Oct. 2017.
[25]The Casa Team et al., "CASA, the Common Astronomy Software Applications for Radio Astronomy", Publications of the Astronomical Society of the Pacific **134**, 114501 (2022) 10.1088/1538-3873/ac9642.