



Multi-scale CLEAN Memory Utilisation Challenges at Square Kilometre Array Scale

Daniel Wright^{*(1)}, Karel Adánek⁽¹⁾, and Wesley Armour⁽¹⁾

(1) Oxford e-Research Centre, Department of Engineering Sciences, University of Oxford, 7 Keble road, OX1 3QG, Oxford, United Kingdom

Abstract

Deconvolution is a very important tool in radio astronomy, used to remove the point spread function from the image produced by an interferometer. Currently popular deconvolution algorithms are up to 50 years old and were designed for much smaller instruments than are currently being constructed. At image sizes proposed for the Square Kilometre Array (SKA), in this paper we show that the commonly used msCLEAN algorithm developed by Cornwell requires up to 13TB of memory to successfully run. We detail the steps of the msCLEAN algorithm and the incremental memory usage for each. We also propose some memory reduction techniques to improve performance.

1 Introduction

Radio interferometers used in radio astronomy sparsely sample the frequency domain visibilities due to their limited number of antennas. Because of this, a sampling function, known as the point spread function (PSF), is convolved with the true sky brightness distribution image (I_{Actual}) by the instrument. In order to conduct useful science this PSF must be removed from the final image. The image produced by the interferometer is called the dirty image (I_{Dirty}) and can be represented as the actual image convolved with the PSF plus some noise, shown in (1).

$$I_{\text{Dirty}} = I_{\text{Actual}} * \text{PSF} + \text{noise} \quad (1)$$

Due to the presence of noise, direct deconvolution cannot be used to recover the actual image and a deconvolution algorithm must be used. The most popular is a family of matching pursuit non-linear iterative deconvolution algorithms called CLEAN. These algorithms are now ageing. Högbom CLEAN, first published in 1974 [1] was designed for much smaller interferometers than the current state-of-the-art. When reaching the scale of the SKA these algorithms struggle with high memory usage, restricting the image sizes they can be used with.

2 CLEAN Deconvolution

Högbom CLEAN consists of a loop which finds a maxima within the flux of a dirty image, subtracts a fraction of the PSF from that maxima and adds a fraction of the maxima to a model image at the same coordinates the maxima was

detected at. The fraction used is defined by the loop gain. This loop continues until one of its stop conditions is met, either a maximum number of iterations is reached or a minimum amount of flux for the maxima is detected. Once the loop is completed, the model image is convolved with an idealised telescope beam called the CLEAN beam and any flux left in the dirty image is added to the convolved model, this produces the final deconvolved image. This process was further refined for the deconvolution of extended structures by Cornwell in 2008 [2], known as multi-scale CLEAN (msCLEAN). This is accomplished by scaling the dirty image and PSFs by a number of scaling kernels. The same type of loop as in Högbom CLEAN is used, except it searches for a maximum across all the scaled dirty images and subtracts each of the scaled PSFs from associated scaled dirty images.

3 msCLEAN Implementation

The implementation of msCLEAN studied here is taken from the original paper by Cornwell and his implementation of the algorithm in the Radio Astronomy Simulation, Calibration and Imaging Library (RASCIL) [3] application created for SKA.

An example of an msCLEAN deconvolution follows in this section, with each subsection dealing with a different step in the process. The data used for the example is from an observation taken by the Karl G. Jansky Very Large Array (VLA) in configuration D on 23rd August 2010 of supernova remnant G055.7+3.4 [4]. The data is provided as part of a tutorial, where the recommended settings for processing are given [4]: loop gain = 0.1, threshold = 12mJy, maximum iterations = 1,000 and scales $n = [0, 6, 10, 30, 60]$ pixels. The size of the input dirty image is defined as $S \times S$ pixels, where S is the dimension of each side of the image, in the case of the example data $S \times S$ is 512 x 512 pixels.

3.1 Scale Kernels

Scale Kernels are created for each given scale (K_0, K_1, K_2, K_3, K_4), with the specified width in pixels. A scale of 0 pixels means only a single pixel of unit size is placed in the centre of the kernel. These kernels are commonly formed of 2D Gaussians or prolate spheroids, with prolate spheroids favoured in RASCIL.

Currently, RASCIL creates these kernels at double the dimension of the dirty image, so a dirty image sized slice with the center of the kernel shifted to any position can be taken from them. Memory usage (in bytes), in terms of scales (n), bytes used to represent each number in the array (b) and dirty image dimension (S) is given by (2)

$$Memory = 4S^2 \times n \times b \quad (2)$$

3.2 Scaled PSFs

Scaled PSFs are created by convolving the PSF with each scale kernel twice, in all possible combinations (3). This is because a maximum located at the scale i needs to be subtracted from scaled image j using a relevant PSF, which needs to be scaled by both kernels. [5].

$$PSF_{scaled} = PSF * K_i * K_j \quad (3)$$

For the 5 scale example considered here, this leads to the creation of 25 scaled PSFs, as shown in Figure 1 and because the PSFs are usually twice the dimensions (therefore 4 times the memory size) of the dirty image this step has very high memory usage.

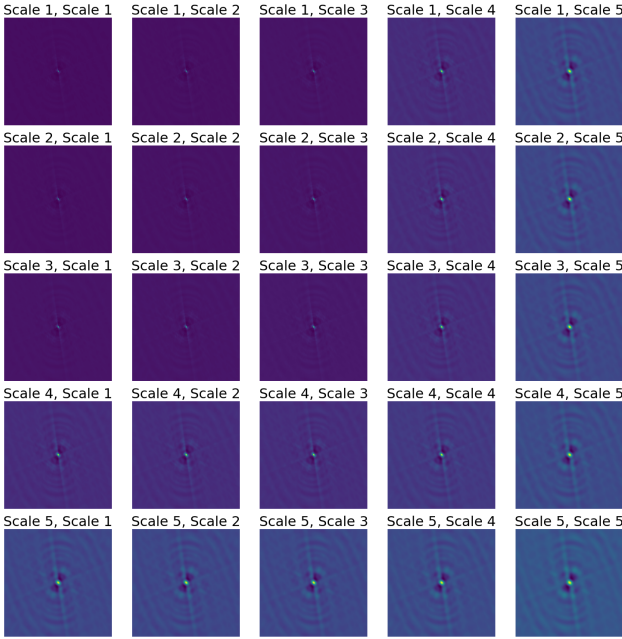


Figure 1. PSFs created for the 5 scale example. 25 PSFs at 1024 x 1024 pixels each

Memory usage is given by (4).

$$Memory = 4S^2 \times n^2 \times b \quad (4)$$

A set of scale bias factors are calculated from the scaled PSFs, by finding the maximum flux within each scaled PSF. The result for a PSF convolved with the same kernel twice is used as the scaling bias factor for that scale. e.g. the maximum in the PSF convolved with the scale 2 kernel twice is the scaling bias factor for scale 2 images. These bias factors allow comparison between scales.

3.3 Scaled Dirty Images

The dirty image is convolved once by each scale kernel to produce a set of scaled dirty images (5). One scaled image is produced for each scale as shown in Figure 2.

$$I_{Scaled} = I_{Dirty} * K_i \quad (5)$$

Memory usage is given by (6).

$$Memory = S^2 \times n \times b \quad (6)$$

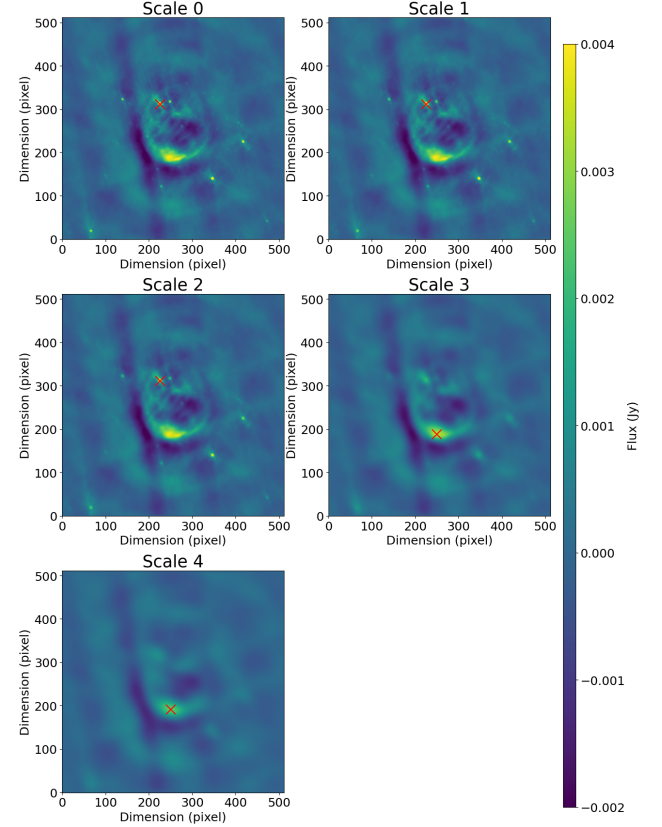


Figure 2. Dirty images created for the 5 scale example, all 512 x 512 Pixels, with maxima marked for each scale.

3.4 CLEAN Loop

The algorithm then proceeds with the various created scaled images to the minor loop.

Step 1: Each scaled dirty image is individually scanned for the position and magnitude of the highest flux level. This gives one position and magnitude result for each scale, each of the maxima are marked in Figure 2.

Step 2: Each of the maxima discovered is biased by dividing by the appropriate scale factor.

Step 3: Once all maxima have been scaled, find the overall maximum and note the scale that produced it. In the case of this example scale 4 has the overall maximum.

Step 4: Check if the maximum from step 3 is below a set threshold, if it is the loop is broken here.

Step 5: The scale kernel for the scale found in step 3 is added to the clean components at the position found for that scale in step 1, Figure 3 shows the slice used for this. The kernel is multiplied by the biased overall maximum from step 3 and the set loop gain. Memory usage to store the CLEAN components is given by (7).

$$Memory = S^2 \times b \quad (7)$$

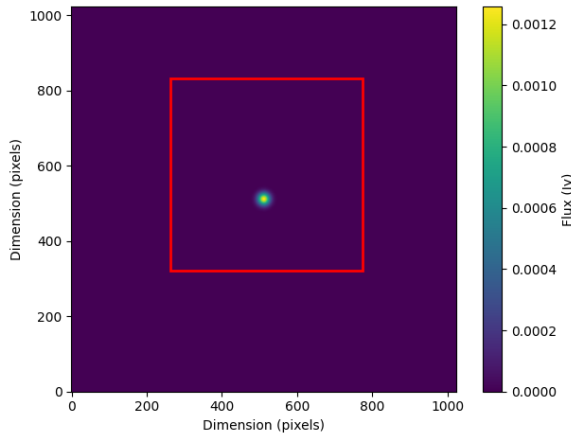


Figure 3. 1024 x 1024 sized scale kernel with 512 x 512 pixel slice marked for addition to the CLEAN map.

Step 6: Find the slice of PSF to overlap and subtract from the dirty image. The PSF is twice the dimension of the dirty image, so to subtract it from the dirty image a dirty image-sized slice must be taken. The slice is taken with the centre of the PSF shifted to the position the overall biased maximum was found at in step 3, in this case scale 4. Similar slices are taken for all PSFs scaled first with kernel 4 and secondly 0, 1, 2, 3 and 4.

Step 7: Subtract the scaled PSF slices from the corresponding scaled dirty images. The example in Figure 4 shows the slice from PSF scale 4, scale 3 to be subtracted from the shown dirty image scale 3. The slice from PSF scale 4, scale 1 is subtracted from dirty image scale 1, continuing in that pattern.

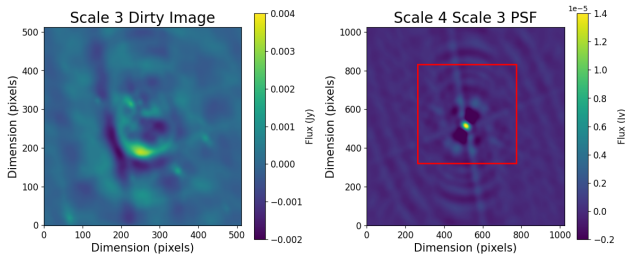


Figure 4. Scale 3 dirty images with corresponding scale 4, scale 2 PSF slice for subtraction.

Step 8: Return to step 1 until the required number of iterations are completed.

3.5 Finalising the Image

Once the required number of loops or the flux threshold has been reached, the algorithm is left with 2 images: the residuals and the CLEAN map. The residuals show the remaining flux left at each scale after cleaning, Figure 5 shows scale 0 residuals. A small amount of structure can be seen left behind in the residuals when following the tutorials recommended number of iterations. Performing more iterations removes more of this structure, but leads to artefacts in the final image due to over-cleaning.

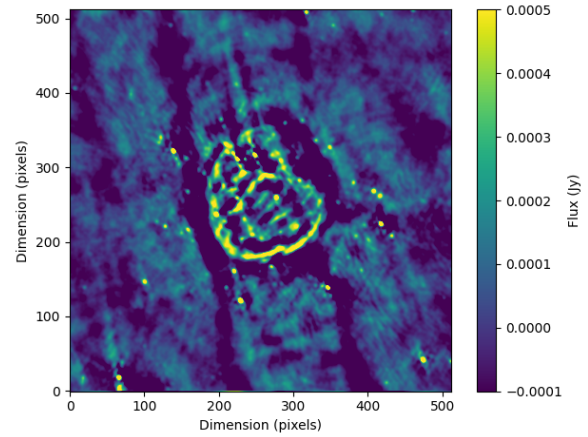


Figure 5. Residual image for scale 0.

The final clean map shows where all the clean components were added throughout the loop. this represents the areas where structures were detected, shown in Figure 6

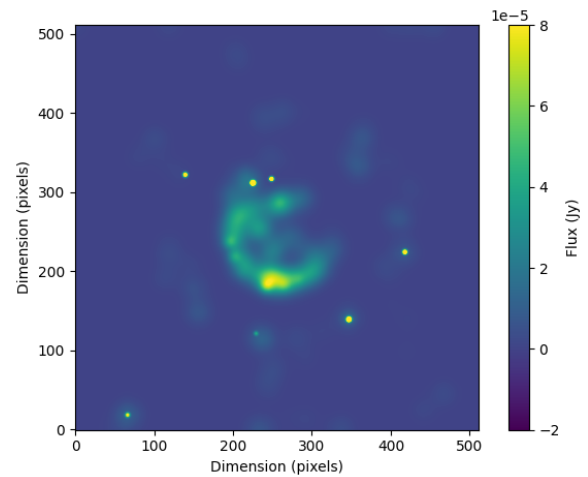


Figure 6. Final CLEAN map.

The final clean map is convolved with an idealised representation of the telescopes beam, called the clean beam. This beam is a Gaussian which is fit to the main lobe of the PSF.

The final deconvolved image is produced by adding the convolved CLEAN map and the scale 0 residuals together.

The result of this example is given in Figure 7.

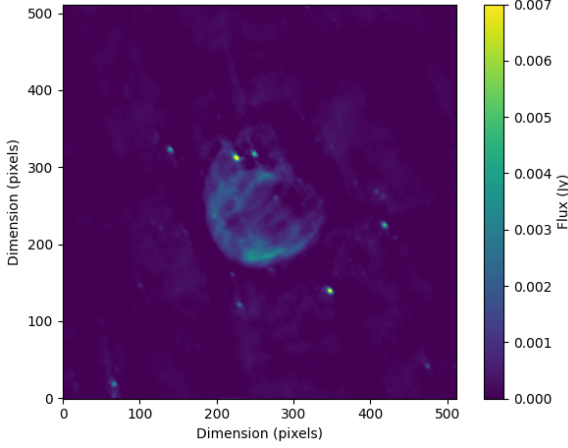


Figure 7. Final deconvolved image.

4 Memory Usage Examples

Current estimates produced by SKA show images are up to 113,204 x 113,204 pixels are expected to be created [6]. We can use the information above and assume double precision arithmetic, to work out the memory usage for images of this magnitude and show how quickly the memory requirements can compound for $n = 5$ scales, in Table 1.

Table 1. Memory Usage for Different Sized Images

| Dimension | Kernels (GB) | Scaled PSF (GB) | Scaled I_{Dirty} (GB) | Components (GB) | Total (GB) |
|-----------|--------------|-----------------|--------------------------------|-----------------|------------|
| 1,000 | 0.16 | 0.8 | 0.04 | 0.008 | 1.008 |
| 5,000 | 4 | 20 | 1 | 0.2 | 25.2 |
| 10,000 | 16 | 80 | 4 | 0.8 | 100.8 |
| 20,000 | 64 | 320 | 16 | 3.2 | 403.2 |
| 50,000 | 400 | 2,000 | 100 | 20 | 2,520 |
| 113,204 | 2,050.4 | 10,252.1 | 512.6 | 101.5 | 12,917.7 |

This shows that at the maximum image size specified for SKA, this method of msCLEAN would require nearly 13TB of memory to execute successfully.

5 Memory Usage Reduction Techniques

For the set of scale kernels (Figure 3) memory savings could be made by not holding such large arrays in memory for small scales. In this example 1024 x 1024 pixel arrays are used even though the largest kernel is only 60 pixels wide. The Kernel arrays should be dynamically sized to fit the pixel size needed with little excess. Alternatively, if the kernels are relatively small, as they are in this example they could be calculated "on-the-fly" and not saved in memory at all.

In the grid of scaled PSFs (Figure 1) memory can be saved due to the commutative nature of convolution demonstrated in (8).

$$PSF * K_1 * K_2 = PSF * K_2 * K_1 \quad (8)$$

So, for example, the scale 1, scale 2 PSF could be repurposed as the scale 2, scale 1 PSF, allowing fewer images to be created. For the example in this paper using this idea could reduce the number of scaled PSFs from 25 to 15, reducing memory usage in this step by 40%.

6 Conclusion

The memory usage for the Cornwell msCLEAN algorithm as realised in RASCIL quickly compounds to very high levels, with multiple terabytes of memory required. The number of scaled PSFs produced is by far the biggest consumer of memory, making ideas for reducing this usage the highest priority. The memory requirements mean that this algorithm is restricted to large and expensive compute clusters, thus excluding many potential users. It also precludes any attempt to GPU accelerate the algorithm at this scale, because GPUs have orders of magnitude less memory than what is required.

7 Acknowledgements

This work was supported by the Science and Technology Facilities Council [grant number ST/W001969/1].

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

References

- [1] Högbom, J. A. "Aperture Synthesis with a Non-Regular Distribution of Interferometer Baselines" *Astronomy and Astrophysics Supplement* **15**, June 1974, pp. 417.
- [2] Cornwell, T. J. "Multiscale CLEAN Deconvolution of Radio Synthesis Images" *IEEE Journal of Selected Topics in Signal Processing*, **2**, 5, 2008, pp. 793–801, doi: 10.1109/JSTSP.2008.2006388
- [3] SKAO, "RASCIL: Radio Astronomy Simulation, Calibration and Imaging Library", January 2024, <https://gitlab.com/ska-telescope/external/rascil-main>.
- [4] NRAO, "VLA CASA Imaging-CASA6.2.0", December 2023, url: https://casaguides.nrao.edu/index.php?title=VLA_CASA_Imaging-CASA6.2.0
- [5] Offringa, A. R. and Smirnov, O, "An optimized algorithm for multiscale wideband deconvolution of radio astronomical images" *MNRAS*, **471**, 1, 2017, pp. 301–316, doi: 10.1093/mnras/stx1547.
- [6] SKAO, "SDP system sizing", January 2024, url: https://ska-telescope.gitlab.io/sdp/ska-sdp-par-model/notebooks/SKA1_System_Sizing.html