

Full length article

Multi-GPU maximum entropy image synthesis for radio astronomy

M. Cárcamo ^{a,*}, P.E. Román ^{a,b}, S. Casassus ^c, V. Moral ^c, F.R. Rannou ^a

^a Departamento de Ingeniería Informática, Universidad de Santiago de Chile, Av. Ecuador 3659, Santiago, Chile

^b Center for Mathematical Modeling, Universidad de Chile, Av. Blanco Encalada 2120 Piso 7, Santiago, Chile

^c Astronomy Department, Universidad de Chile, Camino El Observatorio 1515, Las Condes, Santiago, Chile

ARTICLE INFO

Article history:

Received 26 January 2017

Accepted 25 November 2017

Available online 5 December 2017

Keywords:

Maximum entropy

GPU

ALMA

Inverse problem

Radio interferometry

Image synthesis

ABSTRACT

The maximum entropy method (MEM) is a well known deconvolution technique in radio-interferometry. This method solves a non-linear optimization problem with an entropy regularization term. Other heuristics such as CLEAN are faster but highly user dependent. Nevertheless, MEM has the following advantages: it is unsupervised, it has a statistical basis, it has a better resolution and better image quality under certain conditions. This work presents a high performance GPU version of non-gridding MEM, which is tested using real and simulated data. We propose a single-GPU and a multi-GPU implementation for single and multi-spectral data, respectively. We also make use of the Peer-to-Peer and Unified Virtual Addressing features of newer GPUs which allows to exploit transparently and efficiently multiple GPUs. Several ALMA data sets are used to demonstrate the effectiveness in imaging and to evaluate GPU performance. The results show that a speedup from 1000 to 5000 times faster than a sequential version can be achieved, depending on data and image size. This allows to reconstruct the HD142527 CO(6-5) short baseline data set in 2.1 min, instead of 2.5 days that takes a sequential version on CPU.

© 2017 Elsevier B.V. All rights reserved.

1. Introduction

Current operating radio astronomy observatories (e.g. ALMA, VLA, ATCA) consist of a number of antennas capable of collecting radio signals from specific sources. Each antenna's signal is correlated with every other signal to produce samples of the sky image $I(x, y)$, but on the Fourier domain (Candan et al., 2000). These samples $V(u, v)$ are called visibilities and comprise a sparse and irregularly sampled set of complex numbers in the (u, v) plane. A typical ALMA sampling data set contains from 10^4 to more than 10^9 sparse samples in one or more frequency channels.

In the case where $V(u, v)$ is completely sampled, Eq. (1) states the simple linear relationship between image and data:

$$V(u, v) = \int_{\mathbb{R}^2} A(x, y) I(x, y) e^{-2\pi i(ux+vy)} dx dy. \quad (1)$$

Thus the image can be recovered by Fourier inversion of the interferometric signal (Clark, 1999). In this equation, kernel $A(x, y)$ is called the primary beam (PB) and corresponds to the solid angle reception pattern of the individual antennas and it is modeled as a Gaussian function. If the antennas are dissimilar, it is the geometric mean of the patterns of the individual antennas making up each individual baseline (Taylor et al., 1999).

In the real scenario of collecting noisy and irregularly sampled data, this problem is not well defined (Marechal and Wallach, 2009; Chen, 2011). To approximate the inverse problem of recovering the image from a sparse and irregularly sampled Fourier data a process called Image Synthesis (Thompson et al., 2008) or Fourier Synthesis (Marechal and Wallach, 2009) is used. Current interferometers are able to collect a large number of (observed) samples in order to fill as much as possible the Fourier domain. As an example, Fig. 1 shows the ALMA 400 m short baseline sampling for Cycle 2 observation of the HD142527 protoplanetary disk. Additionally, the interferometer is able to estimate data variance σ_k^2 per visibility as a function of the antenna thermal noise (Thompson et al., 2008).

Many algorithms have been proposed for solving the image synthesis problem and the standard procedure is the CLEAN heuristic (Hogbom, 1974). This algorithm is based on the dirty image/beam representation of the problem (Algorithm A), which results in a deconvolution problem (Taylor et al., 1999). CLEAN has been interpreted as a matching pursuit heuristic (Lannes et al., 1997), which is a pure greedy-type algorithm (Temlyakov, 2008). Image reconstruction in CLEAN is performed in the image space using the convolution relationship, and it is therefore quite efficiently implemented using FFTs. This algorithm is also supervised. The user could indicate iteratively in which region of the image the algorithm should focus. However, statistical interpretation of resulting images and remaining artifacts are yet to be described by a well founded theory.

* Corresponding author.

E-mail address: miguel.carcamo@usach.cl (M. Cárcamo).

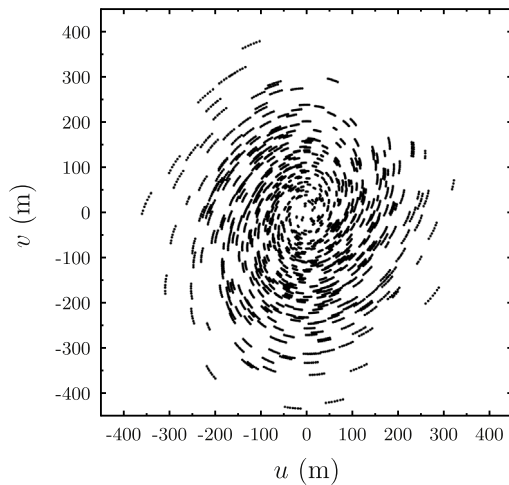


Fig. 1. Short baseline uv sampling of the HD142527 protoplanetary disk.

MEM is inspired by a maximum likelihood argumentation since interferometer measurements are assumed to be corrupted by Gaussian noise (Sutton and Wandelt, 2006; Cabrera et al., 2008). Reconstructed images with this method have been considered to have higher resolution and fewer artifacts than CLEAN images (Cornwell and Evans, 1985; Narayan and Nityananda, 1986; Donoho et al., 1992). MEM was the second choice image synthesis algorithm (Casassus et al., 2006), and is mainly used for checking CLEAN bias and images with complex structure (Neff et al., 2015; Coughlan and Gabuzda, 2013; Warmuth and Mann, 2013). However, routine use of MEM with large data sets has been hindered due to its high computational demands (Cornwell and Evans, 1985; Taylor et al., 1999; Narayan and Nityananda, 1986). For instance, to reconstruct an image of $10^3 \times 10^3$ pixels from the ALMA Long Baseline campaign data set HL Tau Band 3 (279,921,600 visibilities), MEM computes 7.9×10^{16} floating-point operations per iteration, approximately. It was also claimed that MEM could not reconstruct point-source like images over a plateau e.g. Rastorgueva et al. (2011). However, we have found (Section 4) this not to be true using a simulated data set. In fact, the experiments show MEM can reconstruct the point source and maintain a smooth plateau.

The MEM algorithm is traditionally implemented with optimization methods (OM) based on the gradient (Cornwell and Evans, 1985), also called first order methods.

Typical examples of such methods are the conjugate gradient (Press et al., 1992), quasi-newton methods like the modified Broyden's L-BFGS-B (Nocedal and Wright, 2006), and the Nesterov's class of accelerated gradient descent (Nesterov, 2004). All of them require the first derivative calculation which has a computation complexity of $O(M \cdot Z)$, where Z is the number of samples and M is the number of image pixels. The gradient computation is the most expensive part per iteration of first order OMs. In consequence, such OMs are equivalent per iteration from the complexity point of view. In particular, we choose to study the GPU implementation for the Conjugate Gradient (CG). This version uses a positive projected gradient due its simplicity for the case of image synthesis and large data ($Z \sim 10^9$ and $M \sim 10^7$).

A GPU implementation that supports Bayesian Inference for Radio Observations (BIRO) (Lochner et al., 2015) has been proposed before (Perkins et al., 2015). This approach uses Bayesian inference to sample a parameter set representing a sky model to propose visibilities that best match the model. Their model handles point and Gaussian sources. Our approach employs a non-linear optimization problem to directly solve the Bayesian model with a

maximum entropy prior using real and synthetic data. However, our method is not currently able to estimate uncertainties by itself as BIRO does.

In this paper we present a high performance, high-throughput implementation of MEM for large scale, multi-frequency image synthesis. Our aim is to demonstrate computational performance of the algorithm making it practical for research in radio-interferometry for today data sets. The main features of the solution are:

- **GPU implementation:** With the advent of larger interferometric facilities such as SKA (Quinn et al., 2015) and LOFAR (van Haarlem et al., 2013), efficient image synthesis based on optimization cannot be delivered by modern multi-core computers. However, we have found that the image synthesis problem, as formulated by MEM, fits well into the Single Instruction Multiple Thread (SIMT) paradigm (Section 2.6), such that the solution can make efficient use of the massive array of cores of current GPUs. Also, our GPU proposal exploits features like the Unified Virtual Addressing (UVA) and the *Peer-to-peer* (P2P) communications to devise a multiple GPU solution for large multi-frequency data.
- **Parameterized reconstruction:** Supervised algorithms could become obsolete in this high-throughput regime in favor of more automatic methods based on a fitting criteria. Although MEM requires a few parameters, it does not require user assistance during algorithm iteration. In this sense, we say MEM is an unsupervised image synthesis algorithm.
- **Non-gridding approach:** Gridding is the process of resampling visibility data into a regular grid (Taylor et al., 1999). This task is usually carried out by convolving the data with a suitable kernel. However, the resampling process could produce biased results: lose of flux density (Winkel et al., 2016), aliasing and Gibb's phenomenon (Briggs et al., 1999). Although this processing reduces the amount of data and the computation time by data averaging, it does to the detriment of further statistical fit. The use of a high performance implementation allows us to process the full data set without gridding, and still achieve excellent computational performance. However, an interpolation step is still needed to compute model visibilities from the Fourier transform of the image estimate.
- **Mosaic support:** Mosaic images in interferometry allow the study of large scale objects in the sky. In this Bayesian approach we fit a single model image to the ensemble of all pointings. The process of image restoration (see Section 3.3) is then performed on residual images obtained with the linear mosaic formula (Thompson et al., 2008; Taylor et al., 1999).
- **Multi-frequency support:** Spectral dependency can introduce strong effects into image synthesis (Rau and Cornwell, 2011). In this implementation we have applied our algorithm to reconstruct ALMA data sets with several channels and spectral windows, but for a single band. Typically, total bandwidth amounts to a maximum of $\sim 10\%$ the central frequency. In these cases, image synthesis can be performed assuming a zero spectral index (Thompson et al., 2008).

2. Method and implementation

This section describes the mathematical formulation of the mono-frequency *Maximum Entropy Method* (MEM) and the multi-frequency MEM together with a positively constrained conjugate gradient minimization algorithm. Finally, the GPU implementation details are given.

2.1. Data description

Let $V^o = \{V_k^o, k = 0, \dots, Z-1\}$, be the (observed) visibility data, let $\{\sigma_k\} k = 0, \dots, Z-1$ the estimated deviations of each sample, and let $I = [I_i], i = 0, \dots, M-1$. be the image to be reconstructed from V^o . Notice that I is a regularly sampled, and usually a square image function, whereas V^o is a list of sparse points.

Sampled visibilities are grouped in spectral windows. Spectral windows are contiguous spectrum whose frequencies are uniformly spaced and also uniformly divided in channels. Therefore, for the case of multi-spectral, multi-frequency data the observed visibilities are indexed as $V_{w,c,k}^o$ where w corresponds to a spectral window, and c corresponds to a channel of the spectral window w .

Interferometric data also contains polarization. We simplify our approach by considering only the case of intensity polarization (1). This simplification is rather general in practice. Radio interferometers such as ALMA, VLA, or ATCA generate data on orthogonal polarizations (either linear XX, YY, or circular LL, RR), which can enter directly in the algorithm as polarization I e.g. Taylor et al. (1999).

2.2. Bayes formulation and the maximum entropy method

The Maximum Entropy Method (MEM) can be seen as a Bayesian strategy that selects one image among many feasible. Since the data is noisy and incomplete, the solution space is first reduced choosing those images that fit the measured visibilities to within noise level. Among these, the Bayesian strategy selects the one that has a maximum probability of being observed according to a counting rule (Cornwell, 1988; Narayan and Nityananda, 1986; Thompson et al., 2008).

Specifically, let $P(V^o|I)$ be the likelihood of observing the data given image I and let $P(I)$ be a prior knowledge of the image based on the chosen counting rule. Then, by Bayes theorem we obtain the a posteriori probability (Pina and Puetter, 1993)

$$P(I|V^o) = \frac{P(V^o|I)P(I)}{P(V^o)} \quad (2)$$

where $P(V^o)$ is the normalizing constant. Thus, the Maximum a Posteriori (MAP) image estimate is

$$\hat{I}_{\text{MAP}} = \arg \max_I P(I|V^o). \quad (3)$$

\hat{I}_{MAP} corresponds to that image with maximum probability of occurring according to the assumed prior knowledge, among those with maximum probability of producing the observed data within noise level. As exponential functions will be used to model the likelihood, it is preferred to work with the logarithm of Eq. (3). Dropping term $\log P(V^o)$, independent of I , we obtain

$$\hat{I}_{\text{MAP}} = \arg \max_I \Phi(I, V^o) \quad (4)$$

where

$$\Phi(I, V^o) = \log P(V^o|I) + \log P(I). \quad (5)$$

The likelihood $P(V^o|I)$ can be approximated using the fact that visibilities are independent and identically distributed Gaussian random variables corrupted by Gaussian noise of mean zero and standard deviation σ_k . Then, the likelihood can be expressed as

$$P(V^o|I) \propto \prod_k \exp \left\{ -\frac{1}{2} \left| \frac{V_k^m(I) - V_k^o}{\sigma_k} \right|^2 \right\} \quad (6)$$

where $V^m(I)$ denotes model visibilities which are functions of the image estimate.

The MEM image prior is assumed to be a multinomial distribution (Gull and Daniell, 1978) of a discrete total intensity that covers

the entire image, in analogy with a CCD camera array receiving quantized luminance from the sky (Pina and Puetter, 1993). Let M be the number of image pixels and let $N_i = I_i/G$ be the quantized brightness (Sutton and Wandelt, 2006) collected at pixel i . G is considered as the minimal indistinguishable signal variation (Pina and Puetter, 1993). Then, the prior can be expressed by counting equivalent brightness configurations:

$$P(I) = \frac{N!}{M^N \prod_i N_i!} \quad (7)$$

where $N = \sum_i N_i$. Under the assumption of a large number of quanta per pixel the Stirling's approximation ($\log(x!) \sim x \log(x) - x, x \gg 1$) results in Eq. (9). There is a subtle difference from derivation found in Sutton and Wandelt (2006); Pina and Puetter (1993) since the quantization scale G is explicit in the entropy (Cabrera et al., 2008). This results in an additional constraint on the sky intensity $I_i \geq G$, where G is sufficiently small to reproduce a large number of brightness quanta in signal. Taking logarithms to Eqs. (6) and (7), we obtain

$$\log(P(V^o|I)) = -\frac{1}{2} \sum_k \left| \frac{V_k^m(I) - V_k^o}{\sigma_k} \right|^2 \quad (8)$$

$$\log(P(I)) \sim S = -\sum_i \frac{I_i}{G} \log \frac{I_i}{G} + \text{constant}. \quad (9)$$

Replacing Eqs. (8) and (9) into Eq. (5), we arrive at the following objective function for minimization with $\lambda = 1$:

$$\Phi(I, V^o; \lambda, G) = \frac{1}{2} \sum_k \left| \frac{V_k^m(I) - V_k^o}{\sigma_k} \right|^2 + \lambda \sum_i \frac{I_i}{G} \log \frac{I_i}{G}. \quad (10)$$

We recognize Eq. (10) as a typical χ^2 term plus an entropy regularization term $-\lambda S$ with a penalization factor λ as in (11). The previous generalization is a common expression of the objective function for MEM (Narayan and Nityananda, 1986). Other authors argument penalization in Eq. (10) as an entropy-based distance from I to a prior blank image G (Cornwell and Evans, 1985).

$$\Phi(I, V^o; \lambda, G) = \chi^2(V^o, I) - \lambda S(I; G). \quad (11)$$

For multi-frequency data, the χ^2 term is expressed as:

$$\chi^2 = \frac{1}{2} \sum_{w=0}^{W-1} \sum_{c=0}^{C-1} \sum_{k=0}^{Z-1} \left| \frac{V_{w,c,k}^m(I) - V_{w,c,k}^o}{\sigma_{w,c,k}} \right|^2. \quad (12)$$

Finally, the MEM method for image synthesis consists of solving the following non-linear, constrained optimization problem:

$$\hat{I}_{\text{MEM}} = \arg \min_{I \geq G} \Phi(I; V^o; \lambda, G). \quad (13)$$

2.3. MEM parameters

Although our implementation of MEM is an unsupervised algorithm it is still dependent on four important parameters that determine properties of the resulting image. These parameters are the entropy penalization factor λ , the minimal image value G , pixel size Δx , and image size $M^{1/2}$. Even though a complete study on how these parameters affect image properties is beyond the scope of this work, we briefly mention our approach to select reasonable values for them.

In theory, pixel size should satisfy Nyquist criterion, such that image sampling rate should be at least twice as large as the maximum sampled frequency, that is $1/\Delta x \geq 2u_{\text{max}}$. In practice, this is approximated as 1/5 to 1/10 of the Full-Width-Half-Maximum (FWHM) across the main lobe of the dirty beam (see Appendix B).

Assuming the primary beam entirely covers the region of interest, it is recommended that the canvas size ($M^{1/2} \Delta x$) has to be larger than the full extent of the main signal. This argumentation is based on the fact that MEM is known to have optimal performance for nearly black images (Donoho et al., 1992).

In the absence of a non-blank prior image, it is assumed that the minimal value a pixel can take (G) is at most the thermal noise σ_D given by (see Appendix B)

$$\sigma_D = \frac{1}{\sqrt{\sum_k \frac{1}{\sigma_k^2}}}.$$

However, it is preferred to use a small fraction of σ_D , specially in low signal-to-noise regimes. Notice that when (and if) the restored image is computed, an additional factor of σ_D is added to the image.

The penalization factor $\lambda \geq 0$ controls the relative importance between data and the entropy term. When $\lambda = 0$ the problem becomes a least-square optimization problem. In case of a very small lambda the problem is nearly a least-square with a lower bound constrain ($I \geq G$). But when λ increases, less importance is given to data and solution becomes smoother until become a constant equal to G . As long as the image synthesis problem has a non-unique solution, each value for λ represents a possible solution for the problem. Therefore, MEM allows to explore smoother solutions by changing this single parameter at the cost of degrading residuals (χ^2). For example, we start by choosing a very small value of $\lambda \approx 10^{-6}$ which can be considered a near least-square solution. Then, we re-run the program increasing λ by a factor of ten, until the user is satisfied. This procedure can be made practical with a fast enough algorithm implementation.

2.4. Objective function evaluation

The minimization problem (13) can be solved by a constrained Conjugate Gradient (CG) algorithm, which repeatedly evaluates the objective function and its gradient to compute the search direction and step size. Evaluation of the entropy term is straightforward, but evaluation of the χ^2 term requires some additional processing. The following steps are required for computation of model visibilities V^m required for evaluating χ^2 .

1. The attenuation image $A_{m,n}$ or primary beam represents the discrete version of the reception pattern of the telescope in Eq. (1). This image is generally modeled as a Gaussian and depends on the radio-interferometer (Taylor et al., 1999) scaling linearly with the frequency. For ALMA the FWHM is 21 arcsec at 300 GHz for antennas of 12 m diameter.
2. Model visibilities V^F on a grid are obtained by applying a 2D Discrete Fourier Transform to the model image. For the case of multi-frequency data, an attenuation matrix $A_{w,c}$ is built for each channel of each spectral window, resulting in a set of model visibilities, $V_{w,c}^F$. Thus, the Fourier modulation and the interpolation step are carried out independently for each frequency channel.

$$V_{w,c}^F = \mathcal{F}_{2D}\{A_{w,c} \cdot I\}.$$

In case of a mosaic measurement, for each pointing in the sky an attenuation image is calculated and centered in the corresponding field of view.

3. The phase-tracking of the object has a center according to the celestial sphere (Taylor et al., 1999) coordinates. This implies that the image is shifted according to that center. Thus, a modulation factor is applied as follows:

$$\hat{V}^F(u, v) = V^F(u, v) \exp\{2\pi i(u x_c + v y_c)/M^{1/2}\}$$

where (u, v) are the uniformly spaced grid locations, and (x_c, y_c) are the direction cosines of the phase-tracking center. In case of mosaic image, data is divided into several pointing in the sky shifting each field of view according to its location.

4. Model visibilities V^m are approximated from \hat{V}^F by using bilinear interpolation method. Bilinear interpolation has the effect of local smoothing over a cell grid, thus avoiding overshoot effects at object's edges (Burger and Burge, 2010). Aliasing is still an issue in this method, but a common way to overcome this problem is by reducing the Fourier grid cell size increasing the number of image pixels.
5. Residual visibilities $V^R = \{V_k^R\}$ are calculated from $V_k^R = V_k^m - V_k^o$ in order to compute the χ^2 term.

It is worth emphasizing that our algorithm computes the error term for each visibility at their exact uv locations, and do not apply any type of Fourier data gridding to reduce the amount of computation.

2.5. Gradient evaluation

Gradient computation is required for first order optimization methods. Notice that before gradient function is called, the objective function is always executed. Thus, residual visibilities V^R are available for gradient calculation. As well as the objective function the first computation has to do with the entropy gradient. Therefore, every coordinate of the entropy gradient (∇S) has a value that follows Eq. (14).

$$[\nabla S]_i = 1 + \log \frac{I_i}{G}. \quad (14)$$

The χ^2 gradient term is calculated from the derivative of the Direct Fourier Transform of the image (see Appendix C) as shown in Eq. (15). The term $X_i = (x, y)$ is the image coordinate of the i th pixel according to the direction cosines of the phase tracking center, and $U_{w,c,k} = (u, v)$ is the sampling coordinate of the k th visibility at channel c of spectral window s .

$$[\nabla \chi^2]_{m,n} = \sum_{k=0}^{Z-1} \frac{\text{Re}(V_k^R e^{2\pi i X_i \cdot U_k})}{\sigma_k^2}. \quad (15)$$

2.6. SIMT and GPU implementation

The Single Instruction Multiple Thread (SIMT) paradigm is a compute model in which one instruction is applied to several threads in parallel. This means that when a group of threads is ready to run, the SIMT scheduler broadcasts the next instruction to all threads in the group. This is similar but not equal to the Single Instruction Multiple Data (SIMD) paradigm, in which the same instruction is executed over different data, without referring to a particular execution model. This distinction is subtle but important to produce efficient GPU solutions, because an understanding of what is involved in terms of thread execution can make a big difference in the design and performance of a code.

In particular, our approach satisfies the following three general principles for effective and efficient GPU solutions:

1. Write simple and short kernels to use a small number of registers per thread, increasing warp number per Streaming Multiprocessor.
2. Write kernels with spatial locality access to increase memory transfer and cache hit rate.
3. Keep most of the data in global memory to avoid slow transfers from host to device and vice versa.

Although a complete discussion of GPU programming and GPU terminology can be found in Cheng et al. (2014), we briefly discuss here some of the terms used in this paper. A kernel is the code executed by threads in parallel on a GPU. How and when these threads execute on the GPU depend on how the programmer decomposes the problem into sub-problems that can be solved in parallel on the available GPU resources. The most basic GPU resources are the processing elements or cores. A core can run at most one thread at a time. Cores are grouped into one or more Streaming Multiprocessors (SM), and all threads in an SM execute the same instruction at the same time. From the programming point of view, threads are grouped into blocks, which in turn are organized into a grid of blocks. Different thread blocks can be scheduled to run in different SM, but once a block is assigned to a SM, all threads from that block will execute on that particular SM until they finish. This also means that all block resources, like thread registers and shared memory need to be allocated for the entire block execution. Using too many resources per thread will limit the number of thread blocks that can be assigned to a SM, which in turn will limit the number of active blocks running in the SM.

We have implemented a positively constrained CG algorithm to solve Eq. (13). The algorithm's main iteration loop is kept in host, while compute intensive functions are implemented in GPUs. To minimize data transfer, most vectors and matrices are allocated and kept in device global memory. Only the image estimate is transferred back and forth between host and device. At convergence, model visibilities are also sent back to host to compute final residuals.

As an example, here we show implementation details of the two most compute intensive functions, namely Φ and $\nabla\Phi$. Algorithm 1 shows the χ^2 host function that invokes 1D and 2D kernels to accomplish its goal. Lines 2 to 5 apply correction factors and the Fourier transformation steps. Although not shown here, each kernel invocation has an associated grid on which threads are run. For instance, the attenuation and modulation kernels on line (2) and (4), respectively, use a $M^{\frac{1}{2}} \times M^{\frac{1}{2}}$ kernel grid, while the interpolation kernel at line 5 uses a 1D grid of T threads, where T is the next higher power of 2, greater than or equal to Z . All 2D grids are organized in blocks of 32×32 threads, while 1D grids are organized in blocks of 1024×1 threads. The last two steps correspond to the residuals vector computation (line 6) and its global reduction sum (line 7).

Algorithm 1 χ^2 host function

```

1: Chi2( $\tilde{I}, \mathbf{V}^o, \omega, x_c, y_c, \Delta u, \Delta v$ )
2:  $\mathbf{I}_a = \mathbf{KAttenuation}(\tilde{I}, \mathbf{A})$ 
3:  $\mathbf{V}^F = \mathbf{Kcudaffft}(\mathbf{I}_a)$ 
4:  $\hat{\mathbf{V}}^F = \mathbf{KModulation}(\mathbf{V}^F, x_c, y_c)$ 
5:  $\mathbf{V}^m = \mathbf{KInterpolation}(\hat{\mathbf{V}}^F, u/\Delta u, v/\Delta v)$ 
6:  $\mathbf{C} = \mathbf{KChi2Res}(\mathbf{V}^m, \mathbf{V}^o, \omega)$ 
7: KReduce( $\mathbf{C}$ )

```

Algorithm 2 χ^2 1D Kernel

```

1: KChi2Res( $\mathbf{V}^m, \mathbf{V}^o, \omega$ )
2:  $k = \text{blockDim}.x * \text{blockIdx}.x + \text{threadIdx}.x$ 
3: if  $k < Z$  then
4:    $\mathbf{V}_k^R = \mathbf{V}_k^m - \mathbf{V}_k^o$ 
5:    $\chi^2(k) = \omega(k) \cdot (\text{Re}(\mathbf{V}^R(k))^2 + \text{Im}(\mathbf{V}^R(k))^2)$ 
6: end if

```

Algorithm 2 shows the **KChi2Res** kernel. First, each thread computes its index into the grid (line 2), and then computes the corresponding contribution to the χ^2 term (lines 4 and 5).

Algorithm 3 lists the pseudo-code for the calculation of the gradient from Eq. (15), which is the most compute intensive kernel. This kernel uses a $M^{1/2} \times M^{1/2}$ grid. The first lines (2 and 3) compute the index into the 2D grid. Then every thread calculates their corresponding contribution to a cell of the gradient, for every visibility. Recall that sine and cosine functions are executed in the special function units (SFU) of a GPU. Therefore, if Tesla Kepler GK210B has 32 SFU and 192 cores per SM, an inevitable hardware bottleneck occurs.

Algorithm 3 $\nabla\chi^2$ 2D Kernel

```

1: KGradChi2( $\mathbf{V}^R, \omega, x_c, y_c, \Delta x, \Delta y$ )
2:  $j = \text{blockDim}.x * \text{blockIdx}.x + \text{threadIdx}.x$ 
3:  $i = \text{blockDim}.y * \text{blockIdx}.y + \text{threadIdx}.y$ 
4:  $x = (j - x_c) \cdot \Delta x$ 
5:  $y = (i - y_c) \cdot \Delta y$ 
6:  $\nabla\chi^2(N \cdot i + j) = 0$ 
7: for  $k = 0$  to  $Z - 1$  do
8:    $\nabla\chi^2(N \cdot i + j) += \omega(k) * [\text{Re}(\mathbf{V}^R(k)) \cdot \cos(2\pi \langle (u_k, v_k), (x, y) \rangle) - \text{Im}(\mathbf{V}^R(k)) \cdot \sin(2\pi \langle (u_k, v_k), (x, y) \rangle)]$ 
9: end for

```

2.7. Multi-GPU strategy

The main idea of using several GPUs comes from Eq. (12). Since the contribution of every channel to the χ^2 term can be calculated independently from any other channel, it is possible to schedule different channels to different devices, and then do the sum reduction in host. Generally, the number of channels is larger than the number of devices and it is therefore necessary to distribute channels as equal as possible to achieve a balanced workload.

Kernels executing on 64-bits systems and on devices with compute capability 2.0 and higher (CUDA version higher than 4.0 is also required) can use *Peer-to-peer* (P2P) (Cheng et al., 2014) to reference a pointer whose memory pointed to has been allocated in any other device connected to the same PCIe root node. This, together with the Unified Virtual Addressing (UVA) (Cheng et al., 2014), which maps host memory and device global memory into a single virtual address space can be combined to access memory on any device transparently. In other words, the programmer does not need to explicitly program memory copies from one device to another. However, since memory referencing under P2P and UVA has to be done in the same process, OpenMP is used (OpenMP Architecture Review Board, 2015) to create as many host threads as devices to distribute channels.

Fig. 2 illustrates the multi-GPU reconstruction strategy for the case of data with four spectral windows and four channels each, on a four device system. A master thread creates a team of four worker threads which are assigned channels in a round-robin fashion with portions of one channel at a time. Channel data is copied from host to the corresponding device by the master thread. Algorithm 4 shows the multi-frequency host function pseudo code that implements this idea. In line 2 a global variable is defined for the χ^2 term, and in line 3 the number of threads is set, always equal to the number of devices. Each thread invokes the **Chi2** host function (Algorithm 1) with the appropriate channel data, in parallel. Once the function is done, a critical section guarantees exclusive access to workers to update the shared variable. Since the update consists of adding a scalar value to the global sum, we have decided to keep this computation on host.

Computation of $\nabla\chi^2$ follows the same logic as before (see Algorithm 5). The major difference now is that the $\nabla\chi^2$ is a vector allocated on GPU number zero, and a kernel is used to sum up partial results. Recall that with P2P and UVA, kernels can read and write data from and to any GPU.

Algorithm 4 Host multi-frequency χ^2 function

```

1: ParChi2( $\tilde{I}$ )
2:  $\chi^2 = 0$ 
3: set_num_threads(NDevices)
4: #pragma omp parallel for
5: for  $i = 0$  to TOTALCHANNELS - 1 do
6:   cudaSetDevice( $i\%NDevices$ )
7:    $\chi_i = \text{Chi2}(\tilde{I}, \mathbf{V}_i^o, \omega_i, x_c, y_c, \Delta u, \Delta v)$ 
8:   #pragma omp critical
9:    $\chi^2 = \chi^2 + \chi_i^2$ 
10: end for

```

Algorithm 5 Host multi-channel $\nabla\chi^2$ function

```

1: KParGradChi2( $\mathbf{V}^R$ )
2:  $\nabla\chi^2 = 0$ 
3: set_num_threads(NDevices)
4: #pragma omp parallel for
5: for  $i = 0$  to TOTALCHANNELS - 1 do
6:   cudaSetDevice( $i\%NDevices$ )
7:    $\nabla\chi_i^2 = \text{KGradChi2}(\mathbf{V}_i^R, \omega_i, x_c, y_c, \Delta x, \Delta y)$ 
8:   #pragma omp critical
9:   KSumGradChi2( $\nabla\chi^2, \nabla\chi_i^2$ )
10: end for

```

3. Experimental settings

In this section, we test our GPU version of MEM with three synthetic and three ALMA observatory data sets. We use them to demonstrate the effectiveness of the algorithm and to measure computational performance. As we pointed out in the introduction, this paper is not meant to be a comparison between MEM and CLEAN. This task would require an extensive imaging study which is beyond the scope of this work. Nevertheless, we find useful to display CLEAN images as a reference and because it is the most commonly method used today.

3.1. Data-sets

A 1024×1024 simulated image of a 7.5×10^{-4} (Jy/beam) point source located at the image center and surrounded by a plateau was used to show that MEM can indeed reconstruct isolated point sources. Image pixel size was set to 0.003 arcsec and the plateau's magnitude was of 3.9×10^{-4} (Jy/beam). Synthetic interferometer data was created with `ft` task from CASA, and the HL Tauri uv coverage (see Table 1). Image resolution was measured by the Full-Width-Half-maximum (FWHM) for MEM and CLEAN images. For this particular example, $\lambda = 0.1$ was used in MEM.

To further explore image resolution and other properties of MEM, we have created two simulated data sets using task `simobserve`. This task let us sample an input image using parameters such as antenna configurations, source direction, frequency, to name a few.

Firstly, a 512×512 noiseless image of ten point sources distributed as shown in Fig. 3 was simulated at 245 GHz. Point source intensities were randomly assigned normalized values between 0 and 1, and pixel size was set to 0.043 arcseconds. As it can be seen, point sources were located at random positions except for p_4 and p_5 , which were placed 0.215 arcseconds apart.

Secondly, a 256×256 image of a processed version of galaxy 3C288 (Bridle et al., 1989) with a pixel size set to 0.2 arcseconds, shown in Fig. 6(a) was simulated at 84 GHz. The original image is an extended emission radio galaxy at 4.9 GHz with a mix of smooth, low resolution regions together with a small and high-resolution

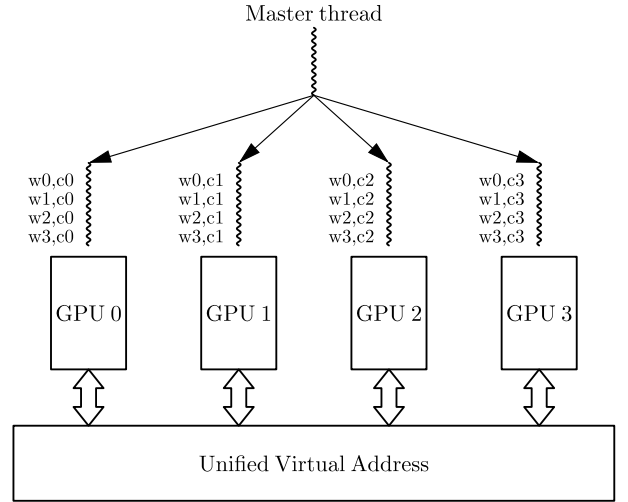


Fig. 2. Illustration of the multi-thread strategy for multi-GPU reconstruction. The figure shows an example of how spectral windows (w_0, w_1, w_2, w_3) and channels (c_0, c_1, c_2, c_3) are distributed with a round-robin scheduling. Data is copied to the UVA before kernel invocation.

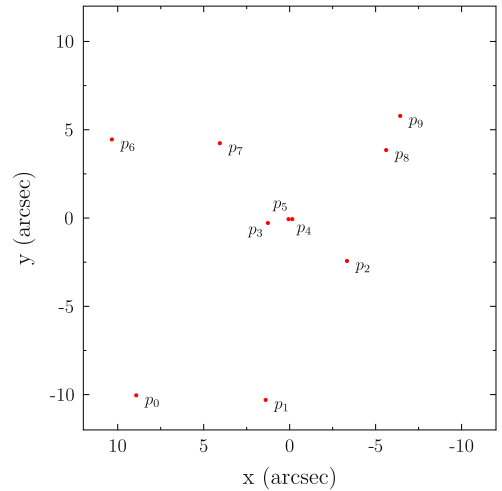


Fig. 3. Point sources image.

filament. Pre-processing implied clipping to zero all pixel values that were smaller than 10^{-3} (Jy/pixel), and adding a 1.0 Jy noise factor to simulated visibilities. Root mean square error (RMSE) of normalized values were measured in a circular region of interest shown in Fig. 6(a).

We test our MEM implementation on real data summarized in Table 1. The software CASA (McMullin et al., 2007) was used to reconstruct CLEAN images that were not downloaded from the ALMA Science verification site and also to use the `mstransform` command to time average specific spectral windows of the HL Tau Band 6 data set.

The first ALMA data set was the gravitationally lensed galaxy SDP 8.1 on Band 7 with four spectral windows, four channels and a total of 121,322,400 non-flagged visibilities (Tamura et al., 2015). Channel 1 of spectral window 3 was completely flagged, so there were no visibilities from this particular channel. MEM images of 2048×2048 pixels were reconstructed, with $\lambda = 0.0$. The CLEAN image was downloaded from the ALMA Science Verification site, and had 3000×3000 pixels.

The second data set was the CO(6-5) emission line of the HD142527 protoplanetary disk, on band 9 with one channel of

Table 1
ALMA data sets summary.

Data set	ADS/JAO.ALMA Code	Beam	SPWs	Channels/SPW	Z
SDP 8.1 Band 7	2011.0.000016.SV	$0.02'' \times 0.02''$	4	4	121,322,400
HLTau Band 6	2011.0.000015.SV	$0.04'' \times 0.02''$	4	4	96,399,248
HLTau Band 6 SPW 0	2011.0.000015.S	$0.03'' \times 0.02''$	1	4	835,360
Antennae North Band 7	2011.0.00003.SV	$1.06'' \times 0.68''$	1	1	149,390
HD142527 CO(6-5) Band 9	2011.0.00465.S	$0.23'' \times 0.18''$	1	1	107,494

107,494 visibilities (Casassus et al., 2015). This small data set was only used for code profiling and measuring GPU occupancy. Larger data sets could not be used to this end because profiling counters (32 bits) overflow with bigger reconstructions.

To test mosaic support the Antennae Galaxies Northern mosaic Band 7 data set was used. This data had 23 fields, 1 channel, and 149,390 visibilities. Once again, images of 512×512 pixels were reconstructed with $\lambda = 0.1$ for MEM reconstruction.

Finally, the HL Tau Band 6 long baseline data set that corresponds to the observations of the young star HL Tauri surrounded by a protoplanetary disk (ALMA Partnership et al., 2015; Pinte et al., 2016; Tamayo et al., 2015). Spectral window zero (four channels) of this data set was used to measure speedup factor for varying image size. Data was time averaged on windows of 300 s, producing only 835,360 visibilities. To stress multi-GPU compute capacity, a MEM image of 2048×2048 pixels was reconstructed from HL Tau full Band 6 with four spectral windows, four channels each and a total of 96,399,248 visibilities. The CLEAN image was downloaded from the ALMA Science Verification site and had 1600×1600 pixels. For reference, Table 1 lists relevant features of all ALMA data used.

3.2. Computing performance metrics

Computational performance was measured by the speedup factor between one GPU and one CPU (single thread) wall-clock execution time versions of MEM, for three images sizes, namely 1024×1024 , 2048×2048 , and 4096×4096 . Since the number of MEM iterations depended on the data set, and some of the sequential reconstructions took excessively long, the average execution time per iteration, was used as a timing measure. All speedup results were for short-spaced channels data sets. The CPU implementation is based on the conjugated gradient function `frrprmn()` from the Numerical Recipes book (Press et al., 1992). It also employs the Fast Fourier transform, line search minimization, and line bracketing functions from the same source. The GPU platform consisted of a cluster of four Tesla K80, each one with two Tesla GK210B GPUs, with 2496 streaming processors (CUDA cores) and 12 Gbytes of global memory. It is important to highlight that this system had two PCI-Express ports on which two GPU were connected to each port. The CPU version was timed on an Intel Xeon E5-2640 V2 2.0 GHz processor.

Unfortunately, speedup cannot be measured in function of GPU cores. Speed factor is usually measured according to a certain number of processing elements like threads or processors. However, GPU programmers do not have control over the number of cores used in every kernel, but only over the grid dimensions. Grid size affects the streaming multiprocessors (SM) occupancy which is defined as:

$$O = \frac{\text{Number of Active Warps}}{\text{Total Number of Warps}}. \quad (16)$$

Thus, occupancy measures how efficiently the SM are being used. Any program inefficiencies like inappropriate grid and block sizes, uncoalesced memory access, thread divergence or unbalanced workload will decrease the number of active warps and therefore SM occupancy and speedup. We used the NVIDIA Pro-

Table 2

Average time per iteration (min) and speed factor for single CPU and single GPU versions of MEM, for two data sets and varying image size.

Image size	Data set	CPU time	GPU time	Speedup
1024 × 1024	CO(6-5)	131	0.08	1638
	HLTau B6w0	2,645	0.60	4408
2048 × 2048	CO(6-5)	349	0.11	3173
	HLTau B6w0	10,339	2.39	4326
4096 × 4096	CO(6-5)	360	0.13	2769
	HLTau B6w0	55,009	9.86	5579

filer (NVIDIA Corporation, 2016b) to measure kernel's occupancy, floating-point operations, compute bound and memory bound kernels.

3.3. Image restoration

Since MEM separates signal from noise (Thompson et al., 2008; Taylor et al., 1999), the CASA package was used to join the MEM model images and their residuals, following the technique of image restoration. Firstly, the MEM model image was convolved with the clean elliptical beam, which effectively changes the MEM image units from Jy/pixel to Jy/beam, and also degrades its resolution. Secondly, a dirty image of each pointing was produced from the visibility residuals, using a user-specified weighting scheme (Briggs et al., 1999). These two last images were added to generate what is called a MEM restored image.

4. Results

4.1. Computational performance

When using CPU timers, it is critical to remember that all kernel launches and API functions are asynchronous, in other words, they return control back to the calling CPU thread prior completing their work. Therefore, to accurately measure the elapsed time it is necessary to synchronize the CPU thread by calling `cudaDeviceSynchronize()` which blocks the calling CPU thread until all previously issued CUDA calls by the thread are completed (NVIDIA Corporation, 2016a).

Table 2 shows average time per iteration in minutes, for the sequential and single GPU versions of MEM. Notice that for the HL Tau case, execution times increased proportionally with image size. Thus, going from a 1024×1024 to a 4096×4096 image, GPU execution time increased by a factor of 16, approximately. A similar behavior can be observed for the CPU case. However, the results for the CO(6-5) data sets are different, in this case the execution time and image size are not proportional. It must be remembered that this is the smallest data set used and neither CPU nor GPU are stressed to their maximal computational capacity. In any case, time differences are significant. A single iteration for the 4096×4096 HL Tau sequential case took 38.2 days to finish, while the same reconstruction in GPU took only 9.85 min. Considering that the algorithm converged in 147 iterations in GPU, it is clear why we could not measure total execution time of the sequential version of MEM. Speedup factors for all cases are presented in Table 2. The smallest and largest speedup factors achieved were 1638 and

Table 3
Occupancy and taxonomy (Gregg and Hazelwood, 2011) of the most important kernels using a 1024 thread block size and different image sizes.

Kernel	Taxonomy	Registers per thread	Achieved occupancy (%)		
			1024 × 1024	2048 × 2048	4096 × 4096
KGradPhi	ND	10	77	76	78
KChi2Res	ND	12	81	79	80
KGradChi2	ND	30	100	100	100
KEntropy	ND	13	86	81	79
KGradEntropy	ND	13	87	81	80
KAttenuation	SDH2D/ND	24	89	90	90
KInterpolation	ND	32	92	90	88
KModulation	ND	18	89	89	88
KReduce	SDD2H	13	75	88	96

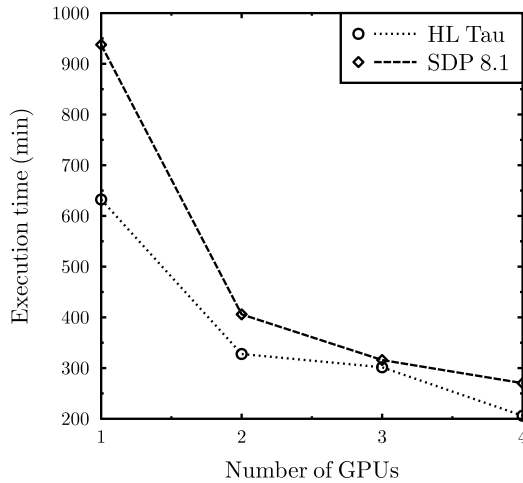


Fig. 4. Average time per iteration using multi-GPU, SDP 8.1 Band 7 and HL Tau Band 6 data sets.

5579, respectively. The smallest speedup corresponded to the short baseline CO(6–5) with the smallest image size. Full reconstruction took 2.5 days in CPU and only 2.1 min in GPU.

Results for multi-GPU reconstruction are displayed in Fig. 4. This graph shows average execution time per iteration as a function of number of GPUs, for the two largest data sets. The total number of visibilities of these data sets were similar, but the number of visibilities per channel were different. Also the SDP 8.1 data had one channel with zero visibilities. Both curves display a typical fixed workload timing behavior, where execution time does not decrease linearly as more GPUs are used. Due to the fact that the number of visibilities in each channel varies, some load imbalance among GPUs may be causing this behavior. However, it is well known in parallel computing that fixed workload always has a maximum speedup possible.

For some of the most important kernels, Table 3 lists number of registers per thread and achieved occupancy for the CO(6–5) data set. The KGradChi2 kernel is the only kernel that achieved 100% occupancy for any image size. This is probably due to the fact that it is the most compute intensive of all the kernels and it is able to maintain the GPU busy without many context-switches. Even though the other kernels require a smaller number of registers per thread than the KGradChi2, which means they can have more active warps per SM, they do not achieve maximum occupancy. Another observation which can be made is that occupancy varies differently for different kernels with image size. For instance, the KGradPhi kernel has a consistent 76% to 78% occupancy, but the KReduce kernel occupancy improves with image size, showing that the kernel itself is not big enough to fully exploit the GPU.

The excellent performance results can be explained by the fact that most implemented kernels do not move data to or from

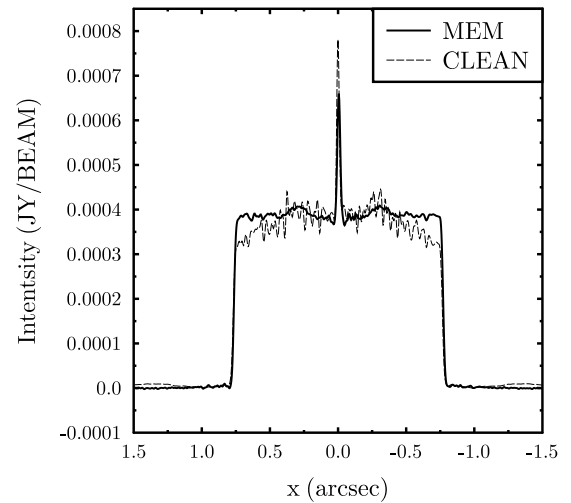


Fig. 5. Reconstructed image profile of a point source in the center of a flat circular plateau, with MEM (restored), $\lambda = 0.1$ and CLEAN. The point source and plateau intensities of the simulated image are 7.5×10^{-4} (Jy/beam) and 3.9×10^{-4} (Jy/beam), respectively.

device. Table 3 also classifies each kernel according to a taxonomy proposed in Gregg and Hazelwood (2011) based on memory-transfer overhead. As it can be seen all kernels except two are of the Non-Dependent (ND) type, which means they do not depend on data transfer between CPU and GPU. The KAttenuation kernel is a Single-Dependent Host-to-Device (SDH2D) type, because it requires data to be moved from host to GPU in order to proceed. But this is only true during the initialization step of the algorithm, after which becomes a ND kernel for the rest of the iterations. For instance, recall Algorithm 2 which is the host function to compute the χ^2 term of the objective function. Once visibility data is on GPU, the program proceeds as a pipeline of kernels which never transfer data back to host, until the last reduce kernel is finished. Nevertheless, the data moved from device to host at the end of each iteration is just a scalar variable and only when the conjugate gradient achieves convergence, the final image and model visibilities are transferred to host.

4.2. Image reconstruction

Fig. 5 depicts CLEAN and restored MEM image profiles of the isolated point source image. On one hand, CLEAN achieved a higher point source intensity, but on the other hand MEM performed a smoother fit of the plateau. The FWHM was 0.064 and 0.062 arcsec for MEM and CLEAN, respectively. Notice that CLEAN overestimated the point source intensity and MEM underestimated the value. CLEAN tends to fit the point source and plateau overestimating the peak, but MEM is known to lose flux in recovered images (Narayan and Nityananda, 1986).

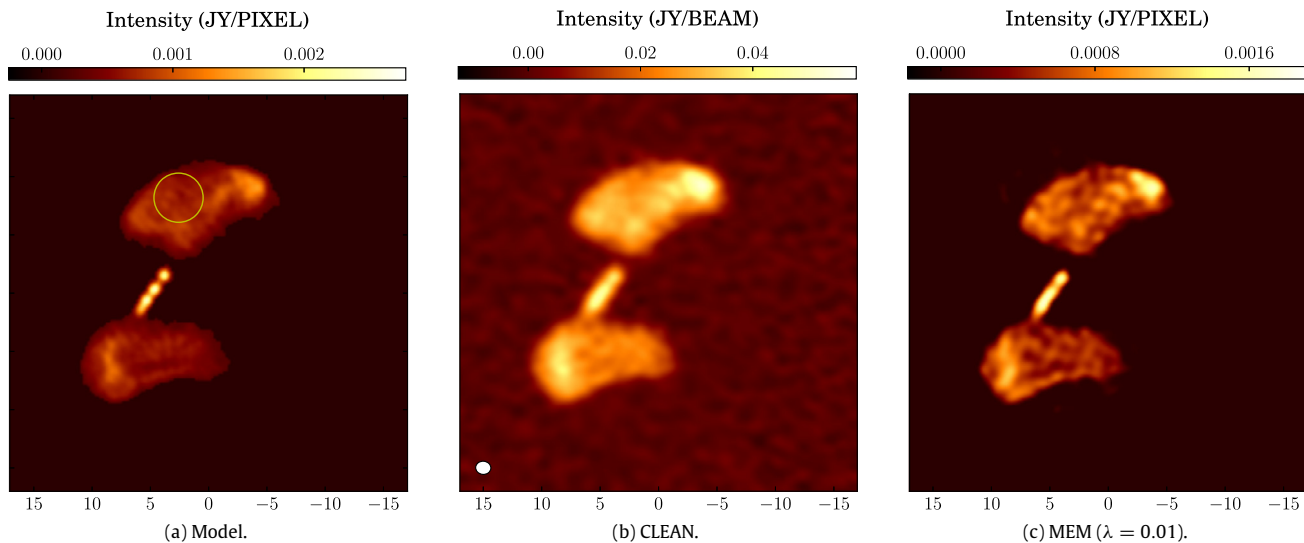


Fig. 6. (a) Model of a modified version of radio galaxy 3C288, (b) CLEAN image (with a beam of $1.31'' \times 1.13''$) and (c) MEM at 84 GHz.

For the field of point sources (Fig. 3), CLEAN delivered uniform resolution across the entire field of view (0.142 arcsecs) and even though MEM ($\lambda = 0.005$) produced better resolution than CLEAN, values greatly depend on image location. For instance, FWHM for p_0 and p_9 were 0.09 and 0.07 arcseconds, respectively, but for point sources near the center like p_3 and p_2 , FWHM was 0.05. Since the distance between the two center points p_4 and p_5 was 0.215 arcsec, resolution at these locations could not be computed in the CLEAN image as there are only 1.5 pixels to fit a Gaussian. MEM instead delivered a FWHM of 0.05 arcseconds for both points. For intensity reconstruction, CLEAN was able to recover over 99.7% of the signal strength in all cases, and MEM recovered differently at different locations. For point sources away from the center, like p_0 and p_1 peak recovery was 98% and 99%, respectively, but for point sources near the center, like p_2 and p_3 , recovery was only 85% and 74%, respectively.

Images for the last simulated data set are displayed in Figs. 6(b) and 6(c). It is clearly noticeable that MEM does a better job removing systematic background noise. In particular at the region of interest, RMSE of normalized reconstructions reach 0.0239 for CLEAN versus a closer 0.0159 for MEM. Filament in the middle of the image is better resolved in MEM due to its smaller resolution. This corresponds to the same phenomena found previously in the point source field. It is worth mentioning that CLEAN images for the first two data sets were reconstructed without user supervision, but for the extended emission example, there were necessary 10 interactive cycles of 1000 iterations each.

Image reconstruction results for real data are shown in Fig. 7. First column displays MEM model images, second column shows MEM restored images, and third column shows CLEAN images. First and second rows corresponds to the SDP 8.1 on Band 7 and HL Tau on Band 6, respectively. As has been said, both are long baseline data sets, and therefore high angular resolution and greater detail are expected. Third row shows the Antennae Galaxies North on Band 7. This data set has short baselines and is also a mosaic. Model images (first column) show images with only signal as MEM separates it from noise, and in all cases restored images are of lower resolution than model images.

5. Conclusions

We have developed a high performance computing solution for maximum entropy image reconstruction in interferometry. The solution is based on GPU for single channel data and on multiple GPUs

for multi-spectral data. The implementation uses a host algorithm that runs the main iteration loop and orchestrates kernel calls. We have decided to write small kernels to improve data locality and minimize thread divergence. Most data is kept on device memory which also minimizes memory moves between host and devices. Overall, we have found that the algorithm renders naturally into a SIMT paradigm and makes it a good candidate for successful GPU implementation.

The resulting code achieves a speedup of approximately 1681 times for our smallest data set, which means that CO(6-5) can be reconstructed in 2.1 min, instead of 2.5 days in single CPU. Long baseline HL Tau Band 6, spectral window zero and 1024^2 image size, takes 58 min approximately. However for full band, multi-spectral data sets, like SDP 8.1 and HL Tau, non-gridded MEM reconstruction is still a challenging task. For instance, full data-set band 6 HL Tau reconstruction requires 200 min per iteration, which is still expensive for routine use with the considered hardware.

Results of image synthesis with MEM has demonstrated that the method removes systematic background noise, has a better resolution, but reaches less intensities at point sources. Moreover, MEM has demonstrate to have less signal variations in flat areas, and having smoother fit than CLEAN on extended structures. This could clearly benefit MEM users by allowing them to identify a finer morphology of the objects, but at the cost of losing signal intensity.

We would like to emphasize that although this work implements the MEM algorithm using the conjugate gradient algorithm, many other first order optimization methods based on gradient can be used. Furthermore, the design permits the use of other well behaved regularization terms to penalize the solution at a similar cost. In consequence we demonstrate that non-linear optimization methods implemented in GPU can be used in practice for research in image synthesis.

Finally, each image synthesis method has its own bias, but making use of many of them could improve the astrophysical analysis of reconstructions. Therefore, a proven high performance MEM implementation could contribute as a tool for image assessment.

The code used in this work is licensed under the GNU General Public License (GNU GPL) v3.0 and is freely available at <https://github.com/miguelcarcamov/gpuvmem>.

6. Future work

Results are encouraging, but there are still several challenges that need to be addressed. Most of the algorithm's time is spent on

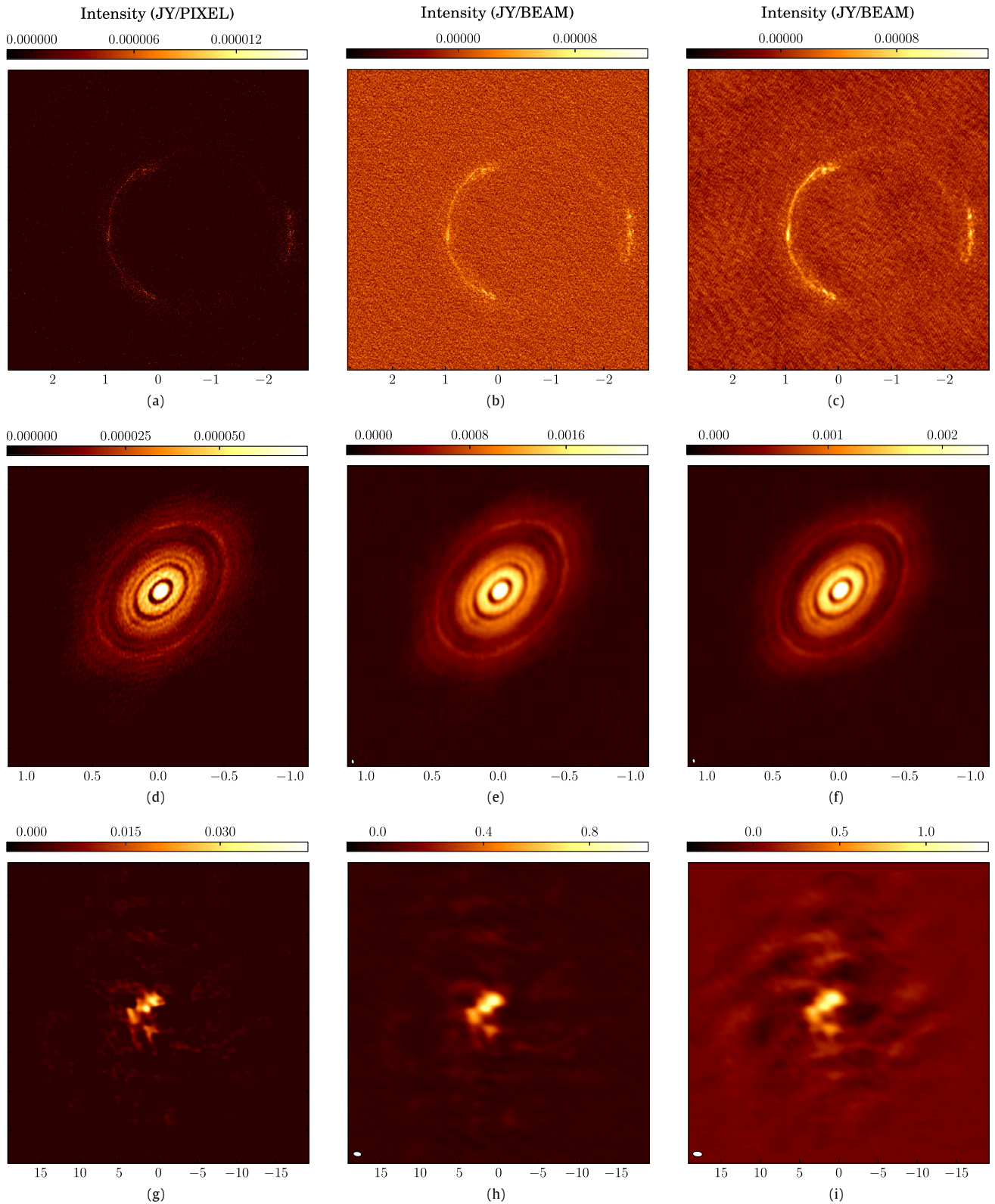


Fig. 7. Image synthesis results using MEM and CLEAN. First column shows MEM model images. Second column shows MEM Restored images, and third column shows CLEAN images. (a)–(c) SDP 8.1 on Band 7, (d)–(f) HL Tau on Band 6. (g)–(i) Antennae Galaxies Northern mosaic on Band 7.

gradient computation, and for nearly-black objects, evaluation on background pixels far from the source carry little or zero contribution to the signal. Therefore, we are currently modifying our code to include a focus-of-attention improvement which will allow the

user to select a region on the image where reconstruction should take place, thus reducing substantially the number of free parameters. We are also working on a solution to reconstruct data sets with long-spaced multiple channels, also called multi-frequency

synthesis. Finally, we are now in a position to conduct a throughout experimental evaluation of MEM imaging parameters including resolution, signal-to-noise ratio, signal recovery, and associated uncertainties for real and simulated data sets.

Acknowledgments

This paper makes use of the following ALMA data: ADS/JAO.ALMA #2011.0.000015.SV, ADS/JAO.ALMA #2011.0.00465.S, ADS/JAO.ALMA #2011.0.00003.SV, ADS/JAO.ALMA #2011.0.000016.SV, ADS/JAO.ALMA #2013.1.00305.S. ALMA is a partnership of ESO (representing its member states), NSF (USA) and NINS (Japan), together with NRC (Canada), NSC and ASIAA (Taiwan), and KASI (Republic of Korea), in cooperation with the Republic of Chile. The Joint ALMA Observatory is operated by ESO, AUI/NRAO and NAOJ. Also, the calculations used in this work were performed in the Brelka cluster, financed by Fondecip project EQM140101 and housed at MAD/DAS/FCFM, Universidad de Chile. F.R. Rannou and M. Carcamo were partially funded by DICYT project 061519RF, Universidad de Santiago de Chile. P. Roman acknowledges postdoctoral FONDECYT projects 3140634, Basal PFB-03 Universidad de Chile, FONDECYT grant 1171841, and Conicyt PAI79160119 Universidad de Santiago de Chile. S. Casasus acknowledges support from Millennium Nucleus RC130007 (Chilean Ministry of Economy), and additionally by FONDECYT grant 1171624.

Appendix A. Beam size

Provided some degree of uv -coverage, a zero-order approximation to the sky signal can be obtained from the dirty image

$$I^D(X) = \sum_k \omega_k V_k^G \exp(2\pi i U_k \cdot X), \quad (\text{A.1})$$

with, $\omega_k = (1/\sigma_k^2) / \sum_l (1/\sigma_l^2)$

where V_k^G are the gridded visibilities (Briggs et al., 1999).

$I^D(X)$ is called the *natural-weights* dirty map and has units of Jy beam⁻¹, where the beam is the solid angle subtended by the FWHM of the Point Spread Function (PSF)

$$\text{PSF}(X) = \sum_k \omega_k \cos(U_k \cdot X). \quad (\text{A.2})$$

The beam is usually approximated by fitting an elliptical Gaussian on the PSF shape and considering the FWHM. The parameters of the ellipse are BMAJ: Major diameter, BMIN: Minor diameter, and BPA: anti-counter clockwise angle in degree. This shape is interpreted as a natural resolution for the CLEAN algorithm.

Appendix B. Thermal noise

Under the assumption of uncorrelated samples¹ we can obtain an analytic expression for the thermal noise σ_D using the following error propagation from dirty image I^D

$$\begin{aligned} \sigma_D^2 &= \text{Var}(I^D) = \text{Var}\left(\text{Re}\left(\sum_k \frac{\omega_k}{\sum_l \omega_l} (V_k^o e^{-2\pi i \theta_k})\right)\right) \\ &= \text{Var}\left(\sum_k \frac{\omega_k}{\sum_l \omega_l} (V_k^{oR} \cos(\theta_k) - V_k^{oI} \sin(\theta_k))\right) \\ &= \sum_k \frac{\omega_k^2}{(\sum_l \omega_l)^2} \left(\frac{1}{\omega_k} \cos^2(\theta_k) + \frac{1}{\omega_k} \sin^2(\theta_k)\right) \end{aligned}$$

¹ Visibility data on uv points closer than the size of the dishes are correlated.

$$\begin{aligned} &= \frac{\sum_k \omega_k}{(\sum_k \omega_k)^2} \\ \sigma_D &= \frac{1}{\sqrt{\sum_k \omega_k}} = \frac{1}{\sqrt{\sum_k \frac{1}{\sigma_k^2}}} \end{aligned}$$

where $\theta_k = U_k \cdot X$, $\omega_k = \frac{1}{\sigma_k^2}$. We also assume uncorrelated imaginary and real part of the samples. σ_D units are Jy beam⁻¹. Then, the constant to change units from Jy beam⁻¹ to Jy pixel⁻¹ is

$$\frac{(\Delta x)^2}{(\pi)/(4 * \log(2)) * \text{BMAJ} * \text{BMIN}}$$

where Δx is the pixel side size in radians.

Appendix C. χ^2 gradient

The gradient of a model visibility is approximated from the DFT of model image.

$$\frac{\partial V_k^R}{\partial I_l} = \frac{\partial V_k^m}{\partial I_l} \sim \frac{\partial}{\partial I_l} \sum_j I_j e^{-2\pi i \theta_{kj}} = e^{-2\pi i \theta_{kl}}$$

where $\theta_{kl} = U_k \cdot X_l$, from previous approximation we obtain

$$\begin{aligned} \frac{\partial}{\partial I_l} \chi^2 &= \frac{\partial}{\partial I_l} \frac{1}{2} \sum_k \frac{\overline{V_k^R} V_k^R}{\sigma_k^2} \\ &= \frac{1}{2} \sum_k \frac{1}{\sigma_k^2} \left(\frac{\partial \overline{V_k^R}}{\partial I_l} V_k^R + \overline{V_k^R} \frac{\partial V_k^R}{\partial I_l} \right) \\ &= \sum_k \frac{1}{\sigma_k^2} \text{Re} \left(\frac{\partial \overline{V_k^R}}{\partial I_l} V_k^R \right) = \sum_k \frac{\text{Re}(V_k^R e^{2\pi i \theta_{kl}})}{\sigma_k^2}. \end{aligned}$$

References

- ALMA Partnership, et al., 2015. The 2014 ALMA long baseline campaign: First results from high angular resolution observations toward the HL tau region. *Astrophys. J. Lett.* 808, L3.
- Bridle, A.H., Fomalont, E.B., Byrd, G.G., Valtonen, M.J., 1989. The unusual radio galaxy 3C 288. *Astrophys. J.* 97, 674–685.
- Briggs, D.S., Schwab, F.R., Sramek, R.A., 1999. Imaging. In: Taylor, G.B., Carilli, C.L., Perley, R.A. (Eds.), *Synthesis Imaging in Radio Astronomy II*. In: *Astronomical Society of the Pacific Conference Series*, vol. 180, pp. 127–149.
- Burger, W., Burge, M., 2010. *Principles of Digital Image Processing: Core Algorithms*. In: *Undergraduate Topics in Computer Science*, Springer, London.
- Cabrera, G.F., Casasus, S., Hirschfeld, N., 2008. Bayesian image reconstruction based on voronoi diagrams. *Astrophys. J.* 672 (2), 1272–1285.
- Candan, C., Kutay, M., Ozaktas, H., 2000. The discrete fractional Fourier transform. *IEEE Trans. Signal Process.* 48 (5), 1329–1337.
- Casassus, S., Cabrera, G.F., Förster, F., Pearson, T.J., Readhead, A.C.S., Dickinson, C., 2006. Morphological analysis of the centimeter-wave continuum in the dark cloud LDN 1622. *Astrophys. J.* 639, 951–964.
- Casassus, S., Marino, S., Perez, S., Roman, P., Dunhill, A., Armitage, P.J., Cuadra, J., Wooten, A., van der Plas, G., Cieza, L., Moral, V., Christiaens, V., Montesinos, M., 2015. Accretion kinematics through the warped transition disk in HD142527 from resolved CO(65) observations. *Astrophys. J.* 811 (2), 92.
- Chen, W., 2011. The ill-posedness of the sampling problem and regularized sampling algorithm. *Digit. Signal Process.* 21 (2), 375–390.
- Cheng, J., Grossman, M., McKercher, T., 2014. *Professional CUDA C programming*. In: *EBL-Schweitzer*. Wiley.
- Clark, B.G., 1999. Coherence in radio astronomy. In: Taylor, G.B., Carilli, C.L., Perley, R.A. (Eds.), *ASP Conf. Ser.* 180: *Synthesis Imaging in Radio Astronomy II*. p. 1+.
- Cornwell, T.J., 1988. Radio-interferometric imaging of very large objects. *Astron. Astrophys.* 202, 316–321.
- Cornwell, T.J., Evans, K.F., 1985. A simple maximum entropy deconvolution algorithm. *Astron. Astrophys.* 143, 77–83.
- Coughlan, C.P., Gabuzda, D.C., 2013. Imaging VLBI polarimetry data from active galactic nuclei using the maximum entropy method. In: *European Physical Journal Web of Conferences*, p. 07009.
- Donoho, D.L., Jhonstone, I.M., Koch, J.C., Stern, A.S., 1992. Maximum entropy and the nearly black object. *J. Roy. Statist. Soc.* 54 (1).

- Gregg, C., Hazelwood, K., 2011. Where is the data? Why you cannot debate CPU vs. GPU performance without the answer. In: IEEE International Symposium on Performance Analysis of System and Software. (IEEE-ISPASS), IEEE.
- Gull, S.F., Daniell, G.J., 1978. Image reconstruction from incomplete and noisy data. *Nature* 272, 686–690.
- Hogbom, J.A., 1974. Aperture synthesis with a non-regular distribution of interferometer baselines. *Astron. Astrophys. Suppl. Ser.* 15, 417–426.
- Lannes, A., Anterrieu, E., Marechal, P., 1997. Clean and wipe. *Astron. Astrophys.* 123, 183–198.
- Lochner, M., Natarajan, I., Zwart, J.T.L., Smirnov, O., Bassett, B.A., Oozeer, N., Kunz, M., 2015. Bayesian inference for radio observations. *Mon. Not. R. Astron. Soc.* 450 (2), 1308–1319.
- Marechal, P., Wallach, D., 2009. Fourier synthesis via partially finite convex programming. *Math. Comput. Modelling* 49 (1112), 2206–2212.
- McMullin, J.P., Waters, B., Schiebel, D., Young, W., Golap, K., 2007. Astronomical data analysis software and systems XVI. In: Shaw, R.A., Hill, F., Bell, D.J. (Eds.), ASP Conf. Ser., San Francisco, CA. p. 127.
- Narayan, R., Nityananda, R., 1986. Maximum entropy image restoration in astronomy. *Annu. Rev. Astron. Astrophys.* 24, 127–170.
- Neff, S.G., Eilek, J.A., Owen, F.N., 2015. The complex north transition region of centaurus a: Radio structure. *Astrophys. J.* 802 (2), 87.
- Nesterov, Y., 2004. *Introductory Lectures on Convex Optimization*. Springer-Verlag, New York, NY, USA.
- Nocedal, J., Wright, S.J., 2006. *Numerical Optimization*. Springer-Verlag, New York, NY, USA.
- NVIDIA Corporation, 2016a. *CUDA C Best Practices Guide*. NVIDIA Corporation. Version 8.0.
- NVIDIA Corporation, 2016b. *CUDA Profiler User's Guide*. NVIDIA Corporation. Version 8.0.
- OpenMP Architecture Review Board, 2015. *OpenMP OpenMP application programming interface Version 4.5*.
- Perkins, S., Marais, P., Zwart, J., Natarajan, I., Smirnov, O., 2015. Montblanc: GPU accelerated radio interferometer measurement equations in support of bayesian inference for radio observations. *Astron. Comput.* 12, 73–85.
- Pina, R.K., Puetter, R.C., 1993. Bayesian image reconstruction - The pixon and optimal image modeling. *Publ. Astron. Soc. Pac.* 105, 630–637.
- Pinte, C., Dent, W.R.F., Menard, F., Hales, A., Hill, T., Cortes, P., de Gregorio-Monsalvo, I., 2016. Dust and gas in the disk of HL tauri: Surface density, dust settling, and dust-to-gas ratio. *Astrophys. J.* 816 (1), 25.
- Press, W.H., Teukolsky, S.A., Vetterling, W.T., Flannery, B.P., 1992. *Numerical Recipes in C: The Art of Scientific Computing*, second ed.. Cambridge University Press, New York, NY, USA.
- Quinn, P., Axelrod, T., Bird, I., Dodson, R., Szalay, A., Wicenc, A., 2015. Delivering SKA Science, CoRR abs/1501.05367.
- Rastorgueva, E.A., Wiik, K.J., Bajkova, A.T., Valtaoja, E., Takalo, L.O., Vetukhnovskaya, Y.N., Mahmud, M., 2011. Multi-frequency VLBA study of the blazar S5 0716+714 during the active state in 2004. *Astron. Astrophys.* 529 (A2).
- Rau, U., Cornwell, T.J., 2011. A multi-scale multi-frequency deconvolution algorithm for synthesis imaging in radio interferometry. *Astron. Astrophys.* 532, A71.
- Sutton, E.C., Wandelt, B.D., 2006. Optimal image reconstruction in radio interferometry. *Astrophys. J. Suppl. Ser.* 162 (2), 401.
- Tamayo, D., Triaud, A.H.M.J., Menou, K., Rein, H., 2015. Dynamical stability of imaged planetary systems in formation: application to HL tau. *Astrophys. J.* 805 (2), 100.
- Tamura, Y., Oguri, M., Iono, D., Hatsukade, B., Matsuda, Y., Hayashi, M., 2015. High-resolution ALMA observations of SDP.81. I. The innermost mass profile of the lensing elliptical galaxy probed by 30 milli-arcsecond images. *Publ. Astron. Soc. Japan* 67, 72.
- Taylor, G.B., Carilli, C.L., Perley, R.A. (Eds.), 1999. *Synthesis Imaging in Radio Astronomy II*. In: *Astronomical Society of the Pacific Conference Series*, vol. 180, Astronomical Society of the Pacific, San Francisco.
- Temlyakov, V.N., 2008. Greedy approximation. *Acta Numer.* 17, 235–409.
- Thompson, A., Moran, J., Swenson, G., 2008. *Interferometry and Synthesis in Radio Astronomy*. Wiley.
- van Haarlem, M.P., Wise, M.W., Gunst, A.W., et al., 2013. LOFAR: The low-frequency array. *Astron. Astrophys.* 556, A2.
- Warmuth, A., Mann, G., 2013. Thermal and nonthermal hard X-ray source sizes in solar flares obtained from RHESSI observations. I. Observations and evaluation of methods. *Astron. Astrophys.* 552, A86.
- Winkel, B., Lenz, D., Flöer, L., 2016. Cygrid: A fast Cython-powered convolution-based gridding module for Python. *Astron. Astrophys.* 591, A12.