Journal of Astronomical Instrumentation, Vol. 6, No. 4 (2017) 1750009 (19 pages)

DOI: 10.1142/S225117171750009X

World Scientific
www.worldscientific.com

Accelerated Deconvolution of Radio Interferometric Images using Orthogonal Matching Pursuit and Graphics Hardware

Jonathan Van Belle[1,4], Richard Armstrong[2,5] and James Gain[3,6]
[1]University of Cape Town, Private Bag X3, Rondebosch 7701, South Africa
[2]Square Kilometer Array, Cape Town
SKA SA, 3rd Floor, The Park, Park Road, Pinelands 7405, South Africa
[3]Department of Computer Science, University of Cape Town
Private Bag X3, Rondebosch 7701, South Africa
[4]jdvbelle@gmail.com
[5]armstrong.richard@gmail.com
[6]jgain@cs.uct.ac.za

Received 2017 April 22; Accepted 2017 November 30; Published 2017 December 19

Deconvolution of native radio interferometric images constitutes a major computational component of the imaging process. An efficient and robust deconvolution operation is essential for reconstruction of the true sky signal from measured telescopic data. The techniques of compressed sensing provide a mathematically-rigorous framework within which to implement deconvolution of images formed from a sparse set of nearly-random measurements. We present an accelerated implementation of the orthogonal matching pursuit (OMP) algorithm (a compressed sensing method) that makes use of graphics processing unit (GPU) hardware. We show that OMP correctly identifies more sources than CLEAN, identifying up to 82% of the sources in 100 test images, while CLEAN only identifies up to 61% of the sources. In addition, the residual after source extraction is 2.7 times lower for OMP than for CLEAN. Furthermore, the graphics implementation of OMP performs around 23 times faster than a 4-core CPU.

Keywords: Deconvolution, interferometric imaging, compressed sensing.

1. Introduction

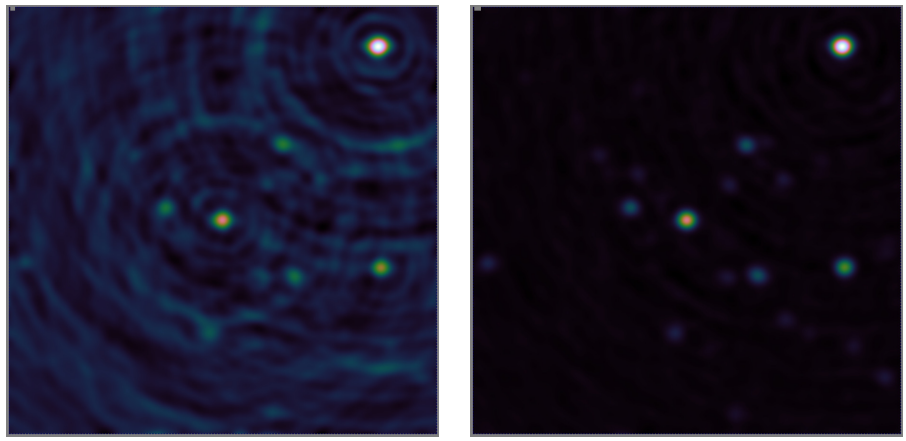
Radio interferometry is a means of observing electromagnetic radiation at radio wavelengths from the sky by using an array of receivers. The multiple receivers in such an array can obtain a resolution nearly equal to that of a single large receiver that encompasses all of the smaller receivers.

Radio interferometers measure the sky through the spatial coherence function, that is, they measure points in the two-dimensional (2D) Fourier domain. These measurements are called *visibilities*, and usually give co-ordinates u and v in a (u, v)-plane. The points they measure correspond to the physical separation of the receivers (baselines). In particular, given a telescope pair with baseline $\mathbf{b}$, the corresponding point measured will be the components of $\mathbf{b}$ orthogonal to the direction of observation (that is the direction from the telescope to the sky source under observation), (u, v). Since it is usually infeasible to have telescopes at every possible baseline, the Fourier domain is therefore sub-sampled.

In addition, since the measured points do not fall on a regular grid, a *Direct Fourier Transform* (DFT) is needed to map them onto the image plane. Since this is often computationally infeasible, the points are typically first mapped onto a regular grid. This process is known as *gridding*. This way, a *Fast Fourier Transform* (FFT) can be used instead of a DFT.

This sub-sampled Fourier plane results in multiple possible images matching a set of measurements taken by an interferometer. One approach to

[1]Corresponding author.



(a) (b)

Fig. 1. Deconvolution of an image: (a) a dirty image and (b) the corresponding deconvolved image.

selecting the best image is to consider all unmeasured visibilities as zero. This results in what is known as the *dirty image*, and suffers from artifacts, as can be seen in Fig. 1(a). Because of the assumption of zero values, the sky's image is convolved with a *Point Spread Function* (PSF), which is the result of the interferometer measuring a single point source (a single bright point in the sky image).

Deconvolution is the process of selecting the image that is the closest to the actual sky from among the possible images. Figure 1(b) shows a deconvolved image using the same measurements. This image is much sparser (that is, more values are zero, or near zero), and is closer to the expected appearance of the sky. It is also much easier to identify the sources in Fig. 1(b) than in Fig. 1(a).

Most deconvolution algorithms attempt to remove this convolution by subtracting the PSF from where the sources are thought to lie (typically the brightest points in the dirty image). However, from our knowledge of the radio sky, we know that most of an image will be background with a few small or point sources, resulting in a sparse image. As such, a good approach to deconvolution is to find the sparsest image that matches the measurements. In order to achieve this we employ *Compressed Sensing* methods.

The theory of *Compressed Sensing* (CS) (Candès, 2006) states that, given a signal x with i non-zero elements, if the measurements can be calculated as a linear combination of x, we have a high probability of being able to reconstruct x with as few as $O(i \log(m))$ measurements, where m is the size of the signal. Orthogonal Matching Pursuit (OMP) is a compressed sensing algorithm which has been shown to be effective at deconvolution of radio astronomy images.

In this work, we adapt OMP to take advantage of several algorithmic optimizations which can be made when reconstructing radio astronomy images. We also accelerate the deconvolution by implementing the algorithm on graphics processors, thereby exploiting the inherent algorithmic parallelism.

Section 2 focuses on the basic theory of Compressed Sensing and OMP, and briefly describes its alternatives. In Sec. 5, we adapt OMP for radio interferometric images. In particular, we make some modifications which reduce the computational complexity of the general form of OMP. We discuss the specifics of implementing OMP on both the CPU and the Graphic Processing Unit (GPU) in Sec. 6, as well as the resource requirements of these systems. We show how well OMP deconvolves a synthetic image, as well as how quickly it performs under various circumstances in Sec. 7.

2. Image Reconstruction with Compressed Sensing

Nyquist–Shannon sampling theory requires that there be at least twice as many measurements as points in the image in order to perfectly reconstruct it. We can, however, use prior knowledge of our image (that it is sparse) to reduce the number of measurements required.

The theory of *Compressed Sensing* (Candès, 2006) shows that compressible signals can be reconstructed with fewer measurements than Nyquist–Shannon sampling theory requires. In particular, the

signal, x, must have a sparse representation in some orthonormal basis. In this case, the N most important co-efficients can be found with only $O(N\log(m))$ measurements, instead of the $2m$ required by the sampling theorem, where m is the highest spatial frequency contained in the signal.

In terms of radio astronomy, this means that we can use CS theory and expect a high probability of reconstructing the image of the actual sky from interferometric measurements so long as the image is sparse enough (has less than half as many non-zero pixels as the number of measurements).

So, given a general linear measurement system

$$\Psi x = y, \tag{1}$$

where x is a vector of the target signal we want to measure (of size n), y a vector of the measurements (of size m), and Ψ an $m \times n$ matrix, which is called the *measurement system*, if $m \ll n$ (there are more unknowns than observations), there are usually multiple solutions for x. We make use of the fact that x is *sparse* in order to find a good solution.

2.1. *Sparsity*

Sparsity is required for the signal, x, to be compressible in some way. This means that there must be a representation of the signal in some space which has fewer entries than its natural representation.

A signal, x, is sparse (respectively weakly sparse) if there is a representation, α, of x which has fewer than n elements. That is, x can be represented by $x = \Phi\alpha$ for some $l \times n$ matrix, with l less than n and other elements are 0 (respectively close to 0). Thus, Eq. (1) becomes

$$\Psi\Phi\alpha = y \quad \text{or} \quad \Theta\alpha = y, \tag{2}$$

where $\Theta = \Psi\Phi$. We call Ψ the *sensing matrix*, and Φ the representation basis.

It is often the case that x already has few non-zero entries, in which case Φ can be taken as the identity matrix (there is no change of space of representation in that case, because x is already sparse in its natural representation space).

2.2. *Incoherence*

Mutual coherence between two matrices, $\mathbf{A}$ and $\mathbf{B}$, is a measure of the largest correlation between any two rows of $\mathbf{A}$ and $\mathbf{B}$, given by

$$\mu(\mathbf{A},\mathbf{B}) = \sqrt{n}\max_{1\le k,j\le n}|\langle \mathbf{A'}_k, \mathbf{B'}_j\rangle|,$$

where $\mathbf{A}'$ contains the rows of $\mathbf{A}$ normalized such that $\langle \mathbf{A'}_i, \mathbf{A'}_i\rangle = 1$.

Compressed sensing requires there to be incoherence between the sensing matrix, Ψ, and the representation basis, Φ. In the case, where the sensing system is the Fourier domain (as is the case in radio interferometry), and if x is already sparse in its natural space, then Ψ is assembled from selected rows of the Fourier transform matrix $\mathbf{F}$, Φ is the identity matrix, and $\mu(\mathbf{F},\Phi) = 1$.

Candes & Romberg (2007) show that a signal can be recovered with high probability when the number of measurements, m, satisfies

$$m \ge C\mu^2(\Psi,\Phi)s\log n,$$

for some constant C, where s is the sparsity of the signal (the number of non-zero entries in its sparse representation).

Lustig *et al.* (2007) have found experimentally that, for incoherent sampling (that is, $\mu(\Psi,\Phi) = 1$), m need only be 2–5 times larger than s in order to have a high probability of reconstructing the signal.

2.3. *Restricted isometry property*

The previous theory focuses on a signal with a sparse representation. However, it is more often the case that most entries will have very small values instead of zero values (for instance, due to instrument noise in the case of radio interferometry). In order to apply CS theory to this case, we need to introduce the Restricted Isometry Property (RIP).

We first define the isometry constant δ_s for each $s \in \{1, 2, \ldots\}$ as the smallest number such that, for all s-sparse vectors α,

$$(1-\delta_s)\|\alpha\|_2^2 \le \|\Theta\alpha\|_2^2 \le (1+\delta_s)\|\alpha\|_2^2.$$

We say that Θ satisfies the RIP of order k if $\delta_k < 1$.

Li *et al.* (2011) show that, if the isometry constant $\delta_{2s} < \sqrt{2}-1$, the ℓ_1 minimization solution, α^*, of Eq. (2) satisfies

$$\|\alpha^* - \alpha\|_2 \le C\frac{\|\alpha - \alpha_s\|_1}{\sqrt{s}}, \tag{3}$$

where α_s is α with all but the largest s entries set to zero, thereby providing a bound on the reconstruction error.

In order to allow for noise in the measurement system, we modify Eq. (2) to

$$\Theta\alpha + E = y,$$

where E represents the measurement error. Equation (3) now becomes

$$\|\alpha^* - \alpha\|_2 \leq C_0 \frac{\|\alpha - \alpha_s\|_1}{\sqrt{s}} + C_1\epsilon, \qquad (4)$$

where ϵ represents a bound on the amount of noise.

Candès (2006) shows that Gaussian measurement matrices, binary measurement matrices, Fourier measurement matrices and incoherent measurement matrices all satisfy the RIP.

2.4. *Basis pursuit*

The direct solution to reconstructing a sparse signal (in the absence of noise) is to solve

$$\min_{\alpha} \|\alpha\|_0 \quad \text{subject to} \quad \Theta\alpha = y,$$

where $\|\alpha\|_0$ is equal to the number of non-zero entries in α.

This is a combinatorial problem (NP Hard) and thus is computationally intractable, so a different approach should be taken.

By providing bounds on the reconstruction error, Eq. (3) shows us that the ℓ_1 norm can provide a good solution (in particular, it provides an upper bound on the reconstruction error). This leads us to the Basis Pursuit (BP) algorithm (Shaobing & Donoho, 1994) by solving

$$\min_{\alpha} \|\alpha\|_1 \quad \text{subject to} \quad \Theta\alpha = y. \qquad (5)$$

In the case of noise in the measurement system, we can use Eq. (4) to obtain the BP De-Noise (BP_ϵ) algorithm by solving

$$\min_{\alpha} \|\alpha\|_1 \quad \text{subject to} \quad \|\Theta\alpha - y\|_2 \leq \epsilon. \qquad (6)$$

By recasting the problem to its unconstrained Lagrangian form, we get the Quadratic basis Pursuit (QP_λ) problem

$$\min_{\alpha}(\|\Theta\alpha - y\|_2^2 + \lambda\|\alpha\|_1), \qquad (7)$$

where λ is a balancing factor between the signal and the noise.

The BP problem can be solved using linear programming, while the BP_ϵ and the QP_λ problems can be solved as second-order cone programs. The QP_λ problem can also be solved using quadratic programming.

2.5. *Matching pursuit*

While ℓ_1 optimization techniques have guaranteed convergence, they often take a large number of iterations to achieve an acceptable solution. *Greedy* algorithms are designed to accurately reconstruct the signal significantly faster.

Algorithm 1 Matching Pursuit

1: $r \leftarrow y$
2: **while** r is significant **do**
3: $\quad p \leftarrow \Theta^{\text{H}} r$
4: $\quad i \leftarrow \arg\max |p_j|$
5: $\quad \alpha_i \leftarrow p_i$
6: $\quad r \leftarrow y - \Theta\alpha$

The Matching Pursuit (MP) algorithm (Mallat & Zhang, 1993), detailed in Algorithm 1, iteratively selects the best element in α which would minimize the residual ($r = y - \Theta\alpha$), and adds it to the solution vector. Since Θ satisfies the RIP, we can choose this element as the element of maximum absolute value in $\Theta^{\text{H}} r$ (Θ^{H} is the conjugate transpose of Θ).

2.6. *Orthogonal matching pursuit*

While MP does converge, in a given iteration the solution vector, α, is not necessarily the optimal solution vector with respect to its non-zero components. The OMP algorithm (Pati *et al.*, 1993), detailed in Algorithm 2, addresses this problem by rebalancing all previously selected values such that the solution vector is optimal for the selected components.

The OMP algorithm operates exactly like MP until the best element is selected. Because we will be weighting this component in each future iteration, its index is stored in an (initially empty) vector, D.

For each selected element, α_i, its contribution to y is determined by the ith row of Θ. As such, the matrix Θ_D is set to be the matrix containing only the columns of Θ whose index is in the set D.

Since each selected element's contribution is perfectly represented by Θ_D, the solution vector of $\arg\min \|y - \Theta_D\alpha\|_2$ will result in the minimum possible residue for the selected components. Finding this solution vector is a well known, solved problem in linear algebra (least-norm solution of underdetermined equations).

Because we have minimized the contribution from each of the selected components, the component selected in the next iteration will not be a result of poor prior weights, resulting in a better component in the next iteration.

Algorithm 2 Orthogonal Matching Pursuit

1: $r \leftarrow y$
2: $D \leftarrow \emptyset$
3: **while** r is significant **do**
4: $\quad p \leftarrow \Theta^{\mathrm{H}} r$
5: $\quad i \leftarrow \arg\max |p_j|$ where $j \notin D$
6: $\quad D \leftarrow D \cup \{i\}$
7: $\quad \alpha \leftarrow \arg\min \|y - \Theta_D \alpha\|_2$ ▷ Only the selected components $\alpha_j, j \in D$ are determined by this problem. Define $\alpha_j = 0$ for $j \notin D$.
8: $\quad r \leftarrow y - \Theta\alpha$

3. Theoretical Comparison to Existing Image Reconstruction Methods

3.1. *CLEAN*

The CLEAN Algorithm (Högbom, 1974) provides a solution to deconvolution by iteratively selecting the brightest point in the dirty image and subtracting a scaled PSF from this point. The scaling factor for this PSF is chosen to be a small portion (usually around 10%) of this brightest point.

The idea is that the brightest point in the image is the most likely place for a source to occur, and by only subtracting a small portion of it, there is less chance of subtracting the residue from some other, nearby source.

The selected points are then considered as the sky model.

One problem with this is that in the case that a bad selection is made, the only way it can be corrected is if, in some iteration, the effects of that selection are brighter than any other point in the image. OMP, on the other hand, rebalances all the chosen weights in each iteration, thereby ensuring that poor selection can be corrected.

3.2. *Maximum entropy method*

The Maximum Entropy Method (MEM) is a deconvolution technique in which we find the image of maximum *entropy* which fits the data to within the noise level. The term entropy (which should not be confused with physical entropy, though the cost function is derived from the physical definition) is something which, when maximized, introduces legitimate *a priori* information. Narayan & Nityananda (1984) find that one of the best entropy forms for general purpose use is

$$F = -\sum_k I_k \ln \frac{I_k}{M_k e},$$

where I_k is the deconvolved image and M_k is a *default* image which allows *a priori* information to be incorporated. A low resolution image of the target can provide a good default image.

The resulting intensity of the model should be consistent with the visibility data, which is measured through a χ^2 statistic. This is the measure of the mean-squared difference between the measurements, V, and the corresponding values for the model, V'

$$\chi^2 = \sum_k \frac{|V_k - V'_k|^2}{{\sigma_k}^2}.$$

Obtaining the solution is an iterative procedure such as CLEAN.

MEM typically performs better on extended structures when compared to CLEAN and CS techniques, while performing worse on point sources (Cotton *et al.*, 1989).

4. Overview of Past and Existing Methods

A large number of deconvolution methods have been developed, often based on particular assumptions about the observed sky. Starck *et al.* (2002) provided a review of many of the algorithms prior to compressed sensing theory, detailing the formalism, strengths and shortcomings of each of them. They make the point that each method is typically based on a particular noise model, or on particular *a priori* assumptions, under which the particular algorithm will perform best.

They expand many of the algorithms into a multi-scale variant using wavelets. By using a particular wavelet basis, they are able to overcome the major shortcomings of each method.

However, a few years later, Candès (2006) introduced Compressed Sensing theory, a method for reconstructing sparse signals using fewer

measurements than are required by the more general Nyquist–Shannon sampling theorem. Since images of the sky are typically sparse, or at least have a sparse representation in some space, this theory can be used for deconvolution.

In addition, Candès (2006) employed the BP algorithm, a method which uses the Compressed Sensing theory to reconstruct a sparse signal.

Tropp *et al.* (2007) provided a description of OMP, a greedy Compressed Sensing technique for general signal recovery. They also provided detailed experimental results on how many measurements are required to recover a signal.

Wiaux *et al.* (2009) provided a detailed explanation of compressed sensing from the perspective of radio astronomy, and constructed a BP algorithm. They then used this algorithm to deconvolve both a dirty image containing point sources and a dirty image containing extended structure, and showed that it produces a superior signal-to-noise ratio to CLEAN.

Schwardt (2012) created a CPU implementation of OMP for radio interferometric data. In addition, he benchmarked the performance of OMP against other compressed sensing algorithms (QP_λ and BP_ϵ), as well as against the Cotton–Schwab CLEAN algorithm. He found that OMP produced sparser images in less time when compared with the other test algorithms, although the image's dynamic range was less than that of the other compressed sensing algorithms.

Schwardt's algorithm executed in 5.3 s for a 100×100 image, executing for 200 iterations using a 2.3 GHz Intel Core i7 CPU. While he did not provide many implementation details, he specified using an FFT, so we can assume it's at least $n \log_2 n$. Extrapolating from this single result implies his implementation (using 200 iterations) would require around an hour for a 4 megapixel image (by contrast, our implementation can manage this in 2.6 s) which, while feasible, is still a long time and can become infeasible for larger images.

Schwardt also found that introducing a positivity constraint on the algorithm (thereby making it identical to Non-Negative Least Squares (NNLS) algorithm described by Lawson & Hanson (1974)) allowed convergence with fewer iterations and produced an image with greater dynamic range. This constraint limits the application to non-negative images however, which is not always the case for radio interferometry.

Carrillo *et al.* (2012) observed that, depending on the observation, astronomical phenomena are typically sparse in either the Dirac basis, wavelet bases, or exhibit gradient sparsity. In particular, astronomical images can contain all these types of structures at once. As such, they define a dictionary of bases in which the image might be sparse, and identify the solution which has the best average sparsity across all chosen bases using a reweighed BP method. They found that this approach produces superior results to methods only optimizing a single basis.

Garsden *et al.* (2015) tested a CS implementation against Cotton–Schwab CLEAN (CoSch-CLEAN) and Multi-Scale CLEAN (MS-CLEAN). They develop a wavelet-based implementation of the FISTA algorithm, a method which replaces the CS ℓ_1 with a smooth approximation, thereby enabling many existing convex-optimization algorithms to solve the compressed sensing problem.

When deconvolving point sources, they found that their CS implementation performed competitively when compared with CoSch-CLEAN. The CS implementation had an improved angular resolution at high and moderate signal-to-noise ratios, with a similar angular resolution at low ratios. The CS implementation was able to detect more faint sources than CoSch-CLEAN, but had a larger error on their flux density values. When deconvolving extended emissions, they found the CS implementation showed an improved angular resolution compared with both CoSch-CLEAN and MS-CLEAN.

Septimus & Steinberg (2010) implemented OMP on a GPU and on FPGAs; however, this only supports images smaller than 128 elements, with a sparsity no greater than 5, which is too restrictive for radio astronomy as it only allows for images of around 11×11 pixels and five non-zero sources.

Fang *et al.* (2011) obtained a 30 to 50 times speedup for their GPU implementation of compressed sensing. Their implementation is not specialized for radio interferometric data, and thus does not take into account the algorithmic enhancements described in Sec. 5. As such, it has a worse algorithmic complexity than can be achieved by a specialized method.

Their GPU implementation executed in 1454 ms for an ";image" of size 16,384 (that is, the equivalent of a 128×128 image), with 4096 measurements and executing for 512 iterations on an

NVIDIA GTX480 GPU. Extrapolating the results for this general algorithm to a megapixel image would result in an execution time somewhere on the order of months, which is again infeasible for most radio astronomy.

5. Adapting OMP to Radio Astronomy

This section will detail the adaptation of the OMP to deconvolution of radio interferometric images. In particular, we are adapting Algorithm 2 to this use case. In terms of compressed sensing, the sensing matrix (Ψ) is a ";masked" 2D Fourier transform matrix, the representation basis (Φ) is the identity matrix, and y is a vector containing the visibilities produced by the interferometer.

This particular choice allows us to make some improvements to OMP. In this section, we detail each step shown in Fig. 2, a diagrammatic overview of OMP.

We initially project the difference by which each candidate component can reduce the residuals. This component is then added to the list of selected pixels. From there, we determine the weights for each selected component that would minimize the residue. This continues until the stopping case is reached. This stopping case is typically a fixed number of iterations.

5.1. *Definitions*

Let y be a vector, of size m, which represents the measurements, and let the image we want to reconstruct be of size n (that is, a $\sqrt{n} \times \sqrt{n}$ image). Let $\mathbf{F}$ be the 2D $n \times n$ Fourier transform matrix (which is multiplied by the size n image vector) and $\mathbf{M}$ the $m \times n$ masking matrix that picks the m measured visibilities from the Fourier plane. Note that $\mathbf{F}^{\mathrm{H}}$ is then the inverse Fourier transform. As such, we have $\Psi = \mathbf{MF}$.

For iteration i, let x_i represent our candidate image, which is initially blank (every entry is zero),

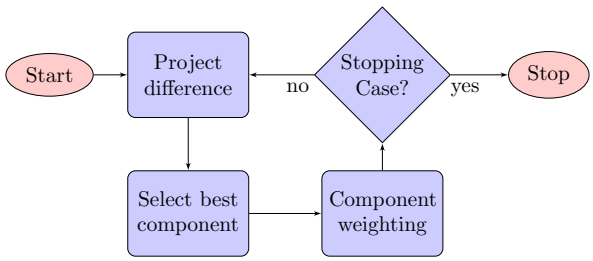


Fig. 2. A diagrammatic overview of OMP.

and let r_i represent the residual image of the candidate x_i (and thus is initially the dirty image). That is, r_i is the difference between the measurements (y) and the measurements that would be obtained from the candidate image ($\mathbf{MF}x_i$), thus $r_i = y - \mathbf{MF}x_i$. In the final formulation, we do not directly need r_i and instead use its Fourier transform. This allows us to avoid the computational cost of de-gridding and re-gridding the image.

Let $\mathbf{P}$ be the matrix representing the 2D convolution by the PSF.

Artifacts are considered to be any features in the dirty image that do not correspond to a convolution of an image by the PSF. These can be caused by noise or as a result of using an approximation in the gridding process.

5.2. *Projection*

The first step of OMP is to identify which column of $\mathbf{MF}$ will most reduce r_i. For column c_i, the projection onto r_i can be computed (according to the general OMP approach) as

$$\frac{c_i \cdot r_i}{\|c_i\|_2}.$$

However, in interferometry we have the property that each entry of c_i is an entry from F and thus $\|c_i\|_2 = \frac{m}{\sqrt{n}}$, a constant factor for any n, and can thus be ignored when identifying the maximum element. Thus we can compute the weighting of each column as the vector

$$(\mathbf{MF})^{\mathrm{H}} r_i = \mathbf{F}^{\mathrm{H}}\mathbf{M}^{\mathrm{H}} r_i.$$

Since we have $r_i = y - \mathbf{MF}x_i$, we get

$$\begin{aligned}\mathbf{F}^{\mathrm{H}}\mathbf{M}^{\mathrm{H}} r_i. &= \mathbf{F}^{\mathrm{H}}\mathbf{M}^{\mathrm{H}}(y - \mathbf{MF}x_i) \\ &= \mathbf{F}^{\mathrm{H}}\mathbf{M}^{\mathrm{H}} y - \mathbf{F}^{\mathrm{H}}\mathbf{M}^{\mathrm{H}}\mathbf{MF}x_i,\end{aligned}$$

where $\mathbf{F}^{\mathrm{H}}\mathbf{M}^{\mathrm{H}} y$ is the dirty image, and $\mathbf{F}^{\mathrm{H}}\mathbf{M}^{\mathrm{H}}\mathbf{MF}x_i$ is x_i convolved with the PSF.

The best column can then be selected by finding the index, i, of the maximum value in $\mathbf{F}^{\mathrm{H}}\mathbf{M}^{\mathrm{H}} r_i$. We can then construct $\mathbf{A}_i$ as the matrix that selects values whose indices correspond to those selected in both the current and in previous iterations (satisfying D in Algorithm 2).

Due to the implementation of gridding and the nature of the radio antennas, artifacts may be produced around the edges of the dirty image (which can be seen in Fig. 4). This can be mitigated by restricting the selection to a window inside the

projection, that is, to not search for the maximum near the boundaries of the image.

5.3. *Weighting*

We now need to identify the weights for the selected columns, x_{i+1}, which best approximate the measurements. We identify this by finding the parameters which minimize the ℓ_2 norm

$$\arg\min_{x_{i+1}} \|y - \mathbf{MFA}_i x_{i+1}\|_2.$$

These can be computed by solving the normal equation

$$(\mathbf{MFA}_i)^{\mathrm{H}}\mathbf{MFA}_i x_{i+1} = (\mathbf{MFA}_i)^{\mathrm{H}} y.$$

General OMP would have us calculate this product directly, however, because of the specific matrices used in interferometry, we can simplify this further. By expanding we get the following:

$$\mathbf{A}_i^{\mathrm{H}}\mathbf{F}^{\mathrm{H}}\mathbf{M}^{\mathrm{H}}\mathbf{MFA}_i x_{i+1} = \mathbf{A}_i^{\mathrm{H}}\mathbf{F}^{\mathrm{H}}\mathbf{M}^{\mathrm{H}} y. \quad (8)$$

Since $\mathbf{M}^{\mathrm{H}}\mathbf{M}$ is a diagonal matrix, it can be considered as a point-wise multiplication by the vector, s, which contains the elements along the diagonal of $\mathbf{M}^{\mathrm{H}}\mathbf{M}$. In particular, point-wise multiplication by s corresponds to applying the sampling function of the observation. Thus $\mathbf{F}^{\mathrm{H}}\mathbf{M}^{\mathrm{H}}\mathbf{MF}$ is simply applying the Fourier transform, the sampling function, and then the inverse Fourier transform. But this is identical to convolving by the PSF, $\mathbf{P}$.

Thus Eq. (8) simplifies to

$$\mathbf{A}_i^{\mathrm{H}}\mathbf{PA}_i x_{i+1} = \mathbf{A}_i^{\mathrm{H}}\mathbf{F}^{\mathrm{H}}\mathbf{M}^{\mathrm{H}} y. \quad (9)$$

Since $\mathbf{F}^{\mathrm{H}}\mathbf{M}^{\mathrm{H}}$ remains fixed between iterations, it can be precomputed and stored. Also, since $\mathbf{P}$ is a 2D-convolution, only the kernel needs to be stored instead of the full matrix.

Furthermore, since $\mathbf{A}_i$ only has a single non-zero value for any column, $\mathbf{A}_i^{\mathrm{H}}\mathbf{PA}_i$ consists simply of scaled values from $\mathbf{P}$ and can be constructed directly (rather than by matrix multiplication).

Between two iterations, exactly one row and one column is added to $\mathbf{A}_i^{\mathrm{H}}\mathbf{PA}_i$. If we solve the system of equations using a matrix inverse, we can thus store the inverse and simply update it on each iteration (the details doing so are shown in Sec. 6.1).

5.4. *Stopping case*

There are several possible stopping cases, depending on what the astronomer requires. The best stopping case, of course, is when the residue is blank and the problem has been perfectly solved. This cannot occur in practice, since there is always inherent noise in the measurements. As such, it is sometimes better to only run for a specific number of iterations, which can be manually tweaked if a first pass produces a bad image. Other options are to stop when further iterations no longer produce much change to the error, or when the error is sufficiently small.

5.5. *Complexity*

For iteration i we have:

(1) The projection step requires a convolution operation, which can be performed using zero-padded FFTs $[O(n \log n)]$, followed by a difference calculation $[O(n)]$, followed by finding the maximum $[O(n)]$. Thus, this step is $O(n \log n)$.
(2) The weighting step requires selecting the values to be added to the matrix $[O(i)]$, followed by updating the matrix inverse $[\mathrm{O}(i^2)]$, and then matrix-vector multiplication $[O(i^2)]$. Thus, this step is $O(i^2)$.
(3) Determining if the stopping case is reached might require computation of the ℓ_2 norm $[O(n)]$.

Thus, when executing for k iterations, the entire deconvolution has a complexity of

$$\begin{aligned} &O\left(\sum_{i=1}^{k}[n \log n + i^2]\right) \\ &\quad = O\left(kn \log n + \frac{k(k+1)(2k+1)}{6}\right) \\ &\quad = O(kn \log n + k^3). \end{aligned}$$

For the most part, the complexity is dominated by the $kn \log n$ part determined by a linear number of Fourier transforms. However, in the case of a large number of iterations (typically many thousands for standard image sizes), the k^3 part may start to dominate, making each successive iteration significantly more costly than the previous one.

6. Implementation

In this section, we discuss the particulars of the implementation. We review the least-squares implementation used in computing the optimal weights for the selected pixels. We also discuss the methods and libraries used in the implementation,

with a focus on how the parallelization was performed for both CPU and GPU architectures. We also look at the expected memory utilization for various image sizes and iteration counts.

6.1. *Least-squares implementation*

In order to obtain the best weights for the selected columns, we need to solve Eq. (9). Normally, we would use a matrix inversion or a Cholesky decomposition in order to solve this, which is a $O(i^3)$ computation. This kind of calculation becomes prohibitively expensive for large matrices.

Fortunately, the matrix we need to invert, $\mathbf{R}_i = \mathbf{A}_i{}^{\mathrm{H}}\mathbf{P}\mathbf{A}_i$, is similar to the matrix already inverted in the previous iteration, $\mathbf{R}_{i-1}$. In particular, since $\mathbf{R}_i$ is constructed as $\mathbf{R}_i = \mathbf{H}_i^{\mathrm{H}}\mathbf{H}_i$ (for $\mathbf{H}_i = \mathbf{M}\mathbf{F}\mathbf{A}_i$) in which $\mathbf{H}_i$ has had a single column, f_i, added to it, we are able to update the inverse obtained in the previous step by using the method described by Fang *et al.* (2011)

$$\mathbf{R}_i{}^{-1} = \left(\begin{array}{c|c} \mathbf{F} & -d\mathbf{R}_{i-1}{}^{-1}\mathbf{H}_{i-1}{}^{\mathrm{H}}c_i \\ \hline -dc_i{}^{\mathrm{H}}\mathbf{H}_{i-1}\mathbf{R}_{i-1}{}^{-1\mathrm{H}} & d \end{array}\right),$$

where

$$d = \frac{1}{\|c_i\|_2{}^2 - c_i{}^{\mathrm{H}}\mathbf{H}_{i-1}\mathbf{R}_{i-1}{}^{-1}\mathbf{H}_{i-1}{}^{\mathrm{H}}c_i},$$

$$\mathbf{F} = \mathbf{R}_{i-1}{}^{\mathrm{H}} + d\mathbf{R}_{i-1}{}^{-1}\mathbf{H}_{i-1}{}^{\mathrm{H}}c_ic_i{}^{\mathrm{H}}\mathbf{H}_{i-1}\mathbf{R}_{i-1}{}^{-1\,\mathrm{H}}.$$

There are a number of repeated calculations in this formulation. To consolidate terms, we let

$$u_1 = \mathbf{H}_{i-1}{}^{\mathrm{H}}c_i.$$

We do not need to explicitly store $\mathbf{H}_{i-1}$ or c_i, as we can obtain this product directly. Since c_i is a column of $\mathbf{MF}$, u_1 is the corresponding row in

$$\begin{aligned}\mathbf{H}_{i-1}{}^{\mathrm{H}}\mathbf{MF} &= (\mathbf{MF}\mathbf{A}_i)^{\mathrm{H}}\mathbf{MF} \\ &= \mathbf{A}_i{}^{\mathrm{H}}\mathbf{F}^{\mathrm{H}}\mathbf{M}^{\mathrm{H}}\mathbf{MF} \\ &= \mathbf{A}_i{}^{\mathrm{H}}\mathbf{P}\end{aligned}$$

and the values can thus be obtained directly from the PSF. We then let

$$u_2 = \mathbf{R}_{i-1}{}^{-1}\mathbf{H}_{i-1}{}^{\mathrm{H}}c_i = \mathbf{R}_{i-1}{}^{-1}u_1.$$

We also note that, when the PSF is considered as the Fourier transform of the Sampling Function, $\|c_i\|_2{}^2$ is simply the zero-frequency component (center) of the PSF.

By substituting u_1 and u_2 into the initial formulation, we then get

$$\mathbf{R}_i{}^{-1} = \left(\begin{array}{c|c} \mathbf{F} & -du_2 \\ \hline -du_2{}^{\mathrm{H}} & d \end{array}\right),$$

where

$$d = \frac{1}{\|c_i\|_2{}^2 - u_1{}^{\mathrm{H}}u_2},$$

$$\mathbf{F} = \mathbf{R}_{i-1}{}^{\mathrm{H}} + du_2u_2{}^{\mathrm{H}}.$$

6.2. *Implementing OMP on the CPU*

Since the FFT dominates the computational complexity, this should be the most rigorously optimized step. Fortunately, a lot of effort has already been expended on optimizing FFTs, and we can thus use the popular FFTW library (Frigo & Johnson, 1998). In particular, this library makes good use of multi-core architectures, as well as the AVX instructions found in most modern desktop CPUs, providing similar performance to vendor-tuned code such as Intel's MKL FFT library (Wang *et al.*, 2014).

Furthermore, since the FFTs are being used to compute a convolution, we need to zero-pad the image to twice its initial height and width in order to compute the discrete convolution (since the convolution theorem applies to the circular convolution). Additionally, because we need to know the contribution that a source on one corner of the image will have in the opposite corner of the dirty image, the convolution kernel will have to extend out to twice the dimension of the dirty image; that is, the PSF will have to extend over twice the horizontal and vertical range of the dirty image. This can then be circularly convolved with the zero-padded image in order to obtain the correct discrete convolution. An example of this is shown in Listing 1. This results in the FFT operating on an image four times the size of the dirty image; this has a significant memory and performance cost. As such, in some cases (when the PSF is sufficiently small and the error is accepted), this might be ignored.

Although the FFT determines the computational complexity, FFTs can be computed very quickly for reasonably sized images (around 50 million floating point operations for a 1 MP image). This means that the various linear steps (finding the maximum element, computing the ℓ_2 norm) become

```
// Copy image to a buffer of twice the height and width.
memset(img_padded, 0, height*width*4*sizeof(*img_pad));
for (int j=0; j<height; ++j)
  for (int k=0; k<width; ++k)
    img_padded[j*width*2 + k] = img[j*width + k];

// Transform to UV co-ordinates via FFT
fftp.fft2(img_padded, uv_padded);

// Point-multiply by a precomputed FFT of the PSF.
// This precomputed mask is correctly shifted and include a normalization factor.
for (int j=0; j<height*2*(width+1); ++j)
  uv_padded[j] *= mask_padded[j];

// Transform back to image co-ordinates via FFT
fftp.ifft2(uv_padded, img_padded);

// Copy image back into original buffer
for (int j=0; j<height; ++j)
  for (int k=0; k<width; ++k)
    img[j*width + k] = img_padded[j*width*2 + k];
```

Listing 1. Convolution performed using FFT to eliminate the cyclical nature of the FFT. A custom FFTW wrapper is used to compute the FFTs.

significant. To compensate, we compute these steps in parallel using the OpenMP framework (Dagum & Menon, 1998).

OpenMP allows us to easily parallelize the linear for loops, so long as each loop iteration is independent from every other iteration. An example of this is shown in Listing 2. In addition, implementing supported reductions is also easily accomplished, an example of which is shown in Listing 3. Performing a more general reduction is somewhat more involved. Listing 4 shows the code used to find the index of the maximum element in an array. For this reduction, each OpenMP thread must accumulate into a separate variable (each of which should be in a separate cache line in order to avoid false sharing (Torrellas *et al.*, 1994)) which are then combined.

A clean window can be chosen, which prevents the selection of an element outside this area in the *projection* step of OMP. This can be used if the sources are known to lie within a particular area, or to prevent selection in the border areas of the image in the case of gridding artifacts.

Since the image resolution is typically higher than the resolving power of the interferometer, the model image is often convolved with a Gaussian beam, called the *restoring beam*, to filter out the higher frequency components. Since CS algorithms can bring super-resolution (the algorithm is able to resolve sources within the angular resolution of the interferometer), whether or not to perform this convolution is currently being debated and, as such, this choice is often more an aesthetic choice rather than a physical choice.

When computing this convolution using the convolution theorem, the Fourier transform of the Gaussian kernel (that is, the restoring beam shape) can be computed in uv coordinates directly (instead of computing the Gaussian kernel in image space,

```
#pragma omp parallel for
for (int i=0; i<width*height; ++i)
  result[i] = lhs[i] - rhs[i];
```

Listing 2. OpenMP code to parellelize a for loop in which each loop iteration is independant from every other iteration (used here to compute a difference image).

```
#pragma omp parallel for reduction(+:sum)
for (int i=0; i<width*height; ++i)
  sum = sum + pow(std::abs(img[i]), 2);
```

Listing 3. OpenMP code for reduction into a sum (used here to calculate the ℓ_2 norm).

```
// Create separate reduction variable (accumulator) for each thread.
//    These variables are separated in memory by the cache line size
//    to prevent false sharing.
int * ix = new int[num_threads * CACHE_LINE_SIZE];
for (int j=0; j<num_threads; ++j)
  ix[j * CACHE_LINE_SIZE] = w_starty*width + w_startx;

// Allow each thread to reduce into a separate accumulator.
#pragma omp parallel for
for (int j=w_starty; j<w_endy; ++j)
  for (int k=w_startx; k<w_endx; ++k)
    if (img[j*width + k] > img[ix[omp_get_thread_num() * CACHE_LINE_SIZE]])
      ix[omp_get_thread_num() * CACHE_LINE_SIZE] = j*width + k;

// Reduce the accumulators to find the index of the maximum element.
int ix_reduce = ix[0];
for (int j=0; j<threads; ++j)
  if (img[ix[j * CACHE_LINE_SIZE]] > img[ix_reduce])
    ix_reduce = ix[j * CACHE_LINE_SIZE];
delete [] ix;
return ix_reduce;
```

Listing 4. OpenMP code for a general reduction (used here to find the index of the maximim element). This is somewhat simpler in later gcc compiler versions, which allow custom OpenMP reductions to be specified.

and then computing the FFT of that kernel), since the Fourier transform of a Gaussian is another Gaussian with an inverted standard deviation. This prevents an unnecessary Fourier transform, and doesn't require any extra memory.

The final implementation requires 19 times the memory required to store the image. This excludes the memory required for initial storage of the image and PSF (an additional five times the storage of the image), the calculation of the inverse matrix ($2k^2 + 2k$ floating point numbers), and a small constant factor. This means that an image requires 96 MB of memory per megapixel for a 32-bit floating point system. While this is more than sufficient for most current interferometers, which produce images smaller than 16 megapixels, future interferometers are expected to produce far larger images. The Square Kilometer Array (expected to be completed by 2024), for instance, is expected to produce up to 10-gigapixel images. Table 1 shows expected memory costs for various image sizes and iteration counts. Table 2 shows the breakdown of the memory cost.

Table 1. Expected memory cost for CPU implementation for various image sizes and algorithm iteration counts.

Image size	1 iteration	1000 iterations	10,000 iterations
1 MP	96 MB	104 MB	896 MB
10 MP	960 MB	968 MB	1.8 GB
100 MP	9.6 GB	9.6 GB	10 GB
1 GP	98 GB	98 GB	99 GB
10 GP	983 GB	983 GB	984 GB

Table 2. Memory cost breakdown for an image of size n and deconvolving for k iterations. Constant factors are not included.

Buffer use	Size
Dirty image	height × width
Deconvolved image	height × width
Residue	height × width
PSF	4 × height × width
Padded image	4 × height × width
FFT of image	4 × height × width
FFT of PSF	4 × height × width
Inverse matrix	k^2
Matrix update	k^2
Solution vectors	$2k$

6.3. *Implementing OMP on the GPU*

The GPU implementation was designed by using the CPU implementation as a template. In particular, the unified memory model introduced in CUDA 6 (Harris, 2013) supports incremental development by allowing each function to be rewritten for the GPU while still compiling into a functioning program.

CUFFT (Nvidia, 2010) was chosen to compute the FFTs, both for its performance and for its similarity to FFTW. In particular, it has a similar API to FFTW, thereby allowing for easier

implementation from the CPU code; it also uses the same input and output format, preventing additional complexities in the code.

The Thrust (Hoberock & Bell, 2009) library was chosen to compute most of the simple computations. This library (which is based on C++'s Standard Template Library) should allow for an optimized implementation for any CUDA device. In particular, Thrust's reduce function is used to select the column in the *projection* step. Since this selection is restricted to a specified window, a custom iterator was implemented which passes over regions outside this window. Thrust was also used to perform simpler operations such as padding the image for the convolution, performing the point-wise multiplication for the convolution, scaling arrays, setting values of array elements, and for permuting arrays.

Similarly, the matrix-vector multiplication has been implemented using CUDA's CUBLAS (Nvidia, 2008) library. One problem with CUBLAS is that it requires a column-major order, while CUFFT requires a row-major order. Fortunately, all the matrices that CUBLAS operates on are symmetric, and thus the order can safely be ignored.

The calculation of the inverse matrix is a bit more complicated:

(1) We need to obtain the u_1 from Sec. 6.1. This is done by simply reading off the appropriate values from the PSF.
(2) We can then use u_1 to obtain u_2 by multiplying u_1 by the inverse matrix obtained in the previous iteration, which is achieved using CUBLAS.
(3) Using Thrust, we can then take the inner product of u_1 and u_2 in order to calculate d.
(4) The previous iteration's inverse matrix must then receive a rank-1 update. Since the result of this update will be the new value in the current iteration, the result is stored directly in a new array allocated for the current iteration's inverse matrix, instead of occurring in-place.
(5) We can then fill in the missing row and column of the new inverse matrix using d and u_2, completing the calculation of the inverse matrix. Since the previous iteration's inverse matrix is no longer needed, that memory can be cleared.
(6) The final step is then to multiply the right-hand side values obtained from the dirty image by the new inverse matrix. This is achieved using CUBLAS.

Table 3. Expected GPU global memory cost for GPU implementation for various image sizes and iteration counts.

	Iterations		
Image Size	1	1000	10,000
1 MP	80 MB	88 MB	880 MB
10 MP	764 MB	772 MB	1.6 GB
100 MP	7.6 GB	7.6 GB	8.4 GB
1 GP	78 GB	78 GB	79 GB
10 GP	778 GB	778 GB	779 GB

For all of these kernels, the block size was determined by CUDA in order to maximize the occupancy. This allows the same code to work on multiple architectures without having to optimize for each one, including future, unknown architectures.

Since the final GPU implementation requires the same memory buffers as the CPU implementation, it requires 19 times the GPU memory required to store the image. In addition, it also requires a small logarithmic factor to perform reductions. This excludes the host (CPU) memory required for the initial storage and final result (7 times the storage of the image), the GPU memory required to store the inverse matrix ($2k^2 + 3k$ floating point numbers), and a constant factor. This means that an image requires about 76 MB of GPU memory per megapixel for a 32-bit floating point system. Table 3 shows expected GPU memory costs for various image sizes and iteration counts.

Additionally, the execution of the GPU kernels is in a separate stream for each image, allowing for multiple images to be deconvolved simultaneously. This can increase the usage of the GPU for small workloads.

7. Results

In this section, we discuss the results obtained by our implementation of OMP. We look at its ability to deconvolve radio interferometric images by deconvolving an image produced using a simulated interferometer and a known sky model. This deconvolved image is compared to the results of the CLEAN algorithm. We also examine the performance characteristics of both our CPU and GPU implementations.

7.1. *Synthetic testing*

One of the major problems in testing imaging systems for radio astronomy is that there is no completely known astronomical image with which to verify the system. As such, we create a synthetic sky model, with randomly positioned point sources, which is then run through a simulation of the KAT-7 interferometer (Jonas, 2009). By knowing the exact sky model, we can easily create a perfect image against which to test. A diagrammatic overview of the comparison process is shown in Fig. 3.

In order to compare the resulting image to a sky model, we need to extract the candidate sources from the deconvolved image. To do this, we run the images through the *Python Blob Detection and Source Measurement* (PyBDSM) (Mohan & Rafferty, 2015) source detection software. This results in a list of candidate sources, which we still need to compare with our sky sources. This provides a residual RMS, σ, that is, the RMS of the remaining image after the sources have been extracted.

While it is sometimes possible to detect structures smaller than the restoring beam (which are called super-resolution methods), such methods often require the observed image to fit particular constraints. We define that two sources are *nearby* to each other if their restoring beams overlap. In particular, we require that the *Full-Width at Half-Maximum* (FWHM) of the restoring beams overlap. We can then create a series of rules by which sources are matched:

- First, we consider sources (in the sky model) that are unresolvable due to a nearby, brighter source (also in sky model). These sources are flagged and will not be considered as matched or missing.
- We consider a source (from the sky model) as successfully matched if there is an extracted source (from the deconvolved image) nearby whose intensity (in Jy/beam) differs by no more than 3σ (which was chosen experimentally, and found to give good results for both CLEAN and OMP). These sources are then flagged as matched. Any unflagged sources (from the sky model) are then considered to be missing (a type II error).
- We also consider an extracted source (from the deconvolved image) as spurious (a type I error) if there is no nearby source whose intensity (in Jy/beam) differs by no more than 3σ.
- Since OMP and CLEAN have a different σ, we also consider the case where the OMP sources are matched to within a difference of CLEAN's 3σ.

In order to ensure that the results are statistically viable, we repeat this process on 100 different sky models.

There are around 30 sources of varying intensity (0–1 Jy) in each model, some of which are outside the 0.5° observation window, resulting in 2660 total sources inside the observation window.

These sky models were run through a simulated interferometer to produce a 1 MP dirty image (including some observation and instrumentation noise). The resulting PSF has a FWHM of around 1′.

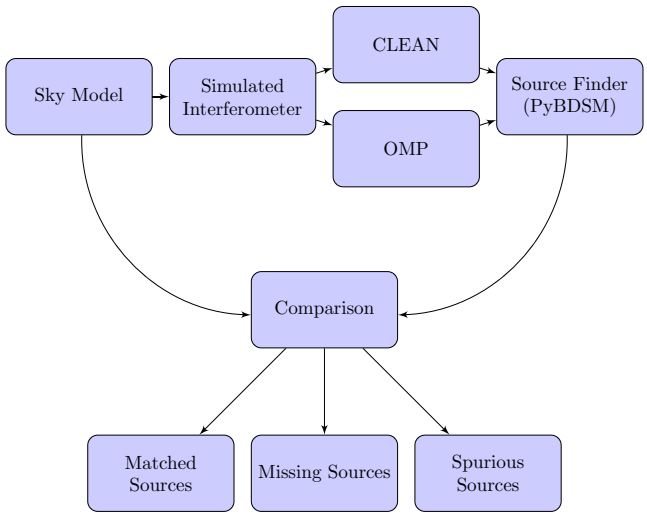


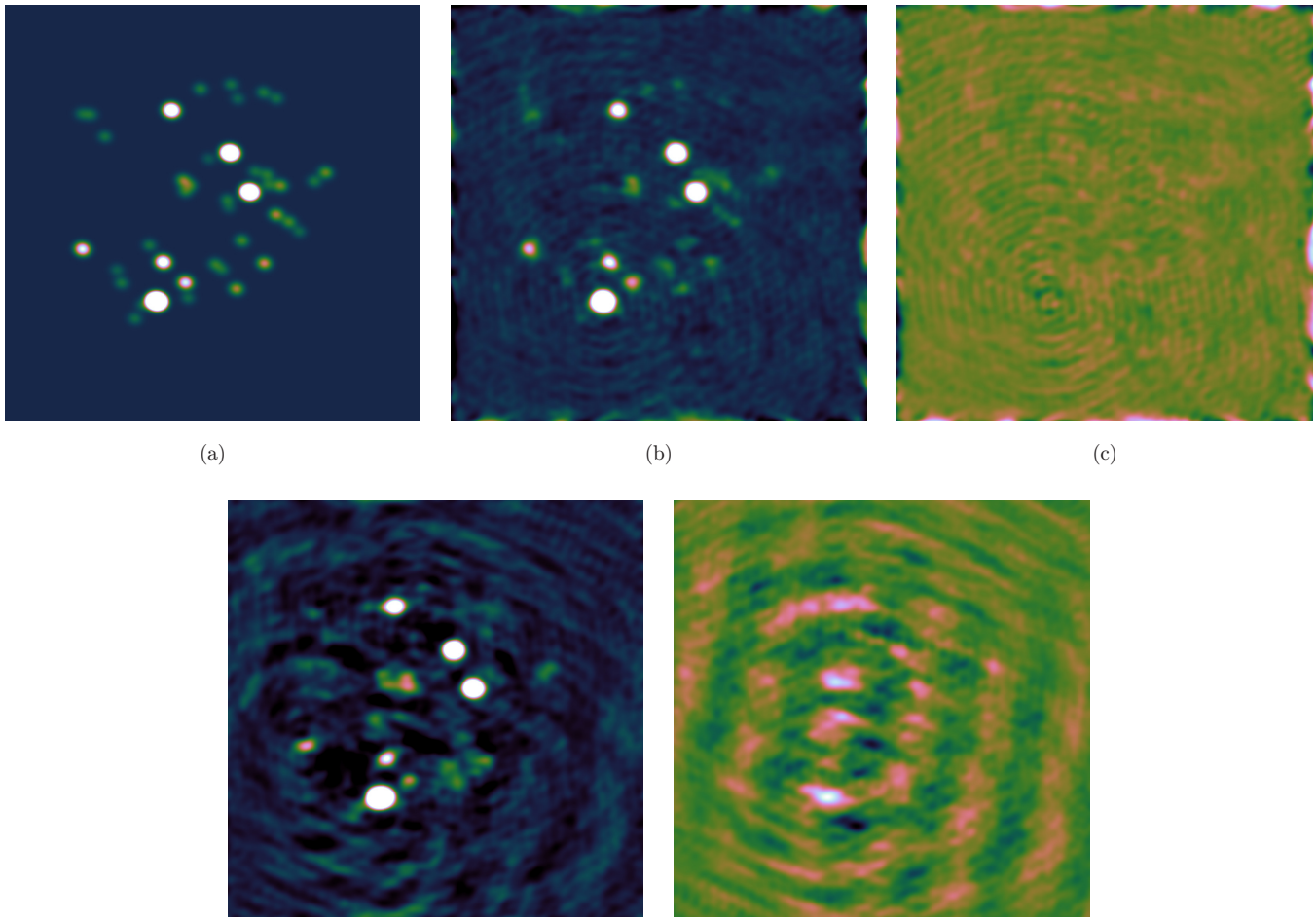
Fig. 3. A simplified diagrammatic overview of the synthetic testing. The sky model is parsed into a simulated interferometer, which produces the dirty image and PSF for that interferometer. Deconvolution is then performed using both CLEAN and OMP, with each producing a deconvolved image. Each of these images is then run through a source finder, and the results are compared with the initial sky model. This comparison will tell us how each deconvolution implementation performed.

7.2. *Comparison to CLEAN*

In order to appropriately test OMP's performance for deconvolution, we will deconvolve synthetic images using both OMP and CASA's Högbom CLEAN, and compare the results.

Additionally, we are interested in the statistical properties of the residual image after deconvolution and source extraction.

Visual inspection of a single image is performed to ensure that the algorithms are operating reasonably, and to get an initial idea of what results to expect. Figure 4 shows the resulting images. It is clear that many of the fainter sources from the sky model, (a), are more easily identifiable in the OMP image, (b), than in the CLEAN image, (d). In



(a) (b) (c)

(d) (e)

Fig. 4. Deconvolution of one synthesized sky model: (a) the sky model convolved with the restoring beam. This is what an ideal interferometer and deconvolution algorithm would produce, (b) the deconvolved image using OMP, (c) the residual after deconvolution using OMP, (d) the deconvolved image using CLEAN and (e) the residual after deconvolution using CLEAN. Images (a), (b), and (d) are scaled to exclude the peak percentile from image (b), while images (c) and (e) are scaled to the full range of image (e).

addition we can see that far less structure remains after deconvolution by OMP, (c), than after CLEAN, (e).

The residual is significantly higher around the edges in images (b) and (c). This is likely due to gridding artifacts resulting in the dirty image not being a perfect convolution of the true sky. These can either be ignored (which may result in some low intensity spurious sources), or they can be removed by initially creating a larger image, which is then cropped after deconvolution. They can also be removed by re-gridding the deconvolved image, as is done in CLEAN, at a significant performance cost.

It should be noted that, for this image, the ℓ_2 norm (which is often used for image comparisons) produced similar results for CLEAN and OMP, with CLEAN slightly outperforming OMP. From looking at the images, however, the results of OMP appear to match the sky model far better than those of CLEAN, with less structure remaining in the residual. As such, we should consider that the ℓ_2 norm might not be a good way to compare interferometric images of point sources.

A more quantitative approach is as follows: 100 synthetic sky models are used to generate dirty images which are then deconvolved using both CLEAN and OMP.

After performing source extraction on both the CLEAN and OMP images and using the comparison described in Sec. 7.1, we get the results shown in Table 4.

OMP significantly outperformed CLEAN, matching 70% (with a standard deviation across all images of $\pm 10\%$) of the 2660 sources contained in all 100 sky models, while CLEAN only matched 45% ($\pm 10\%$) with almost twice an many missing sources.

Table 4. Comparison of CLEAN and OMP. Source counts are summed over all 100 sky models. OMP Relaxed refers to matching sources if they differ by no more than CLEAN's 3σ instead of OMP's.

	CLEAN	OMP	OMP relaxed
Masked sources	378	378	378
Matched sources	1185	1837	1979
Missing sources	1475	823	681
Spurious sources	56	261	169
Spurious sources outside borders	53	224	133
Residual RMS	0.0274	0.0101	0.0101

In addition, the residual RMS noise after sources extraction was 2.7 ($\pm$0.8) times lower for OMP than for CLEAN.

However, OMP produced almost five times the number of spurious sources. A disproportionate number of these spurious sources lie in the border regions where OMP has an increased residual. By cropping out 2% of the image from each side, we reduce this number somewhat (down to 224 spurious sources from 261). This also eliminates three of the spurious sources produced by CLEAN.

7.2.1. *Thresholding OMP and CLEAN equally*

Since OMP results in a significantly lower residual, it also has a lower threshold for source extraction. As such, the feature might result in an extracted source on the OMP image, but not on the CLEAN image. Thus, an important question is whether these extra matched sources, as well as the spurious sources, are purely a result of the lower residual.

As such, a second test was performed by first filtering out any sources below 5$\times$ CLEAN's residual. This is around the level chosen for PyBDSM to consider a feature as valid on the CLEAN images. These results are shown in Table 5.

Table 5. Thresholding at CLEAN's 5σ: comparison of CLEAN and OMP. Source counts are summed over all 100 sky models. OMP Relaxed refers to matching sources if they differ by no more than CLEAN's 3σ instead of OMP's.

	CLEAN	OMP	OMP relaxed
Matched sources	1165	1430	1559
Missing sources	725	460	331
Spurious sources	51	218	126
Spurious sources outside borders	50	215	124

In this case, OMP still outperformed CLEAN by around the same margin, matching 76% ($\pm$11%) of the 1890 sources compared to CLEAN's 56% ($\pm$10%). In addition, this still outperforms the unthresholded CLEAN algorithm, matching 54% of the initial 2660 sources.

In addition, this also eliminates most of the spurious sources outside the border regions.

7.2.2. *Extracted source accuracy*

For this testing, we consider a source to have been matched if its intensity is within a hard threshold, and if the identified position is within a specified distance from the true source. For some applications, it might be important to know how closely the sources resulting from a deconvolution method would match the actual sources.

As such, after matching the sources to the sky model, we performed a test to determine how closely the deconvolved source matches the source in the sky model in both intensity (measured in Jy/beam and in σs) and in distance (measured in restoring beam widths). Given the definitions of the residual RMS and resolving power, we expected the implementations to vary in intensity by about 1σ and in distance by about one beam width.

OMP produced excellent results, with an intensity difference of 0.0083 (with a standard deviation across all matched sources of $\pm$0.0078) Jy/beam, or 0.75σ ($\pm 0.61\sigma$) and a distance of 0.057σ ($\pm 0.103\sigma$) beam widths.

CLEAN, on the other hand, performed poorly when measuring the intensity, with a difference of about 0.040 ± 0.024 Jy/beam, or $1.57\sigma \pm 1.06\sigma$, while performing excellently when measuring the distance, 0.068 ± 0.075 beam widths.

7.2.3. *Two-to-one source matching*

OMP (and, to a lesser extent, CLEAN) will, where possible, optimize sparsity. This will often result in two unresolvable sources in the sky model being represented by a single source in the deconvolved model. In this case, the resulting deconvolved source will have an intensity close to the sum of the two sources, resulting in it being flagged as a false positive, and the two sources in the sky model flagged as unmatched sources.

As such, we modified the source-matching algorithm to consider two sources a match if their

Table 6. Allowing for two-to-one source matching: comparison of CLEAN and OMP. Source counts are summed over all 100 sky models.

	CLEAN	OMP
Matched sources	1594	2139
Missing sources	1066	521
Spurious sources	10	50
Spurious sources outside borders	7	14

intensities are within a factor of two. This will allow for the possibility that up to two unresolvable sources are deconvolved into a single source, or that a single source is deconvolved into two unresolvable sources. The results are summarized in Table 6.

In this case, OMP now outperforms CLEAN by a larger margin, matching 82% (±9%) of the 2660 sources compared to CLEAN's 61% (±11%). In particular, both algorithms now produce far fewer spurious sources, with CLEAN producing only 10 (7 of which lie outside the border region), and OMP producing 50, only 14 of which lie outside the border region.

If we again filter out sources below 5× CLEAN's threshold, OMP matches 92% (±8%) of the 1890 sources compared with CLEAN's 84% (±9%). OMP now only produces slightly more spurious sources than CLEAN, producing 7 spurious sources (5 of which lie outside the border region), compared with CLEAN's 5 spurious sources (4 of which lie outside the border region). These results are summarized in Table 7.

If we also loosen the beam width constraints (such that we consider two sources a match if they are within 1.5 beam widths, instead of 1 beam width), OMP will produce fewer spurious sources than CLEAN. This implies that, for the most part, there is at least a feature near the spurious sources produced by OMP.

Table 7. Allowing for two-to-one source matching and as thresholding at CLEAN's 5σ: comparison of CLEAN and OMP. Source counts are summed over all 100 sky models.

	CLEAN	OMP
Matched sources	1571	1717
Missing sources	319	173
Spurious sources	5	7
Spurious sources outside borders	4	5

7.3. *Runtime performance*

Due to the increasing resolution of interferometers and, in particular, the SKA's large leap in required image size, it is important to accelerate all parts of the interferometry pipeline. Furthermore, near real-time image synthesis allows for more experimentation on the part of the astronomer. Thus, speedup measures form an important part of evaluating a system.

One important consideration is the floating point precision of the program. In particular, Kepler GPUs support two levels of precision, a 32-bit *single precision* floating point type (float), and a 64-bit *double precision* floating point type (double). Since the effect of precision was not known beforehand, both options were implemented. However, on the test dataset, the difference between the resulting image under the two precision types was negligible. As such, single precision was chosen as it executes significantly faster.

Figure 5 shows the base runtime for a single-threaded execution of the implementation for 100 iterations on various image sizes. We can see that the 0.25 MP and 1 MP image take considerably less than a minute to complete execution, with the 4 MP image only requiring slightly more than a minute. The 16 MP and 64 MP images take much longer, with the 64 MP image requiring about 38 min to complete.

In order to improve on this performance, we can make use of multi-core CPUs. As we can see from Fig. 6, a 4-core CPU achieves around a 3.5 times

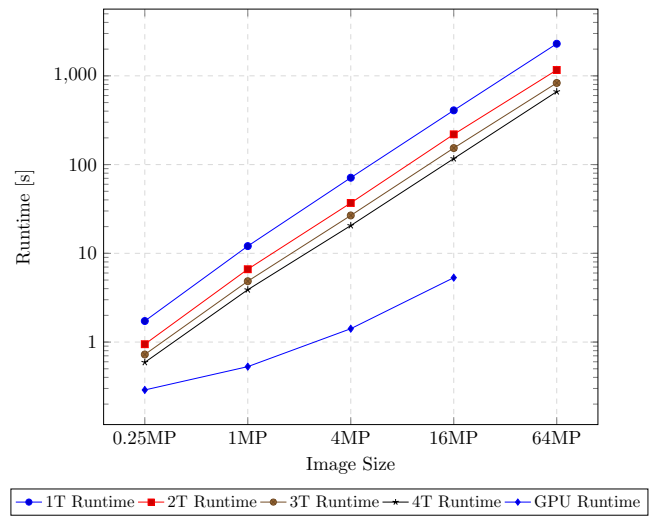


Fig. 5. Runtime for 100 iterations on various image sizes using an Intel i7-4770 and a NVIDIA GTX 770. Note that the axes are logarithmic.

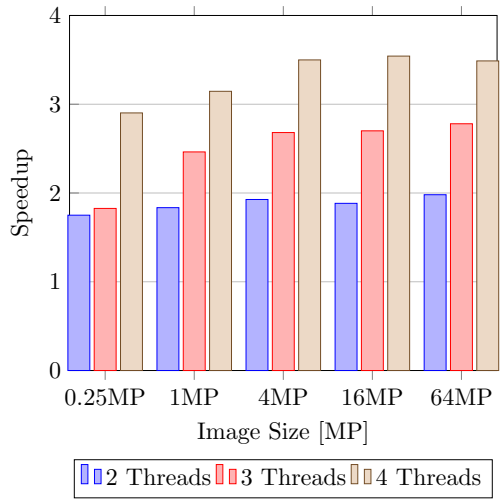


Fig. 6. Speedup obtained on a 4-core CPU (Intel Core i7-4770) for 2, 3 and 4 threads.

speedup for a sufficiently large image. There is no GPU datapoint for 64 MP due to the limited GPU memory of the test system (2 GB).

The speedup for smaller images (smaller than 4 MP) is only relevant if a large number of images need to be processed at once, as their execution time is already sufficiently quick. However, in the case of a large number of simultaneous images, the images can be deconvolved in parallel, allowing for even greater parallelism.

Furthermore, as is shown in Fig. 7, with a NVIDIA GTX770, we can obtain up to an 83 times speedup for a 16 MP image over the single-threaded CPU implementation. This means that a 16 MP image on a GPU only requires around 5 s to deconvolve, compared with the CPU's 400 s.

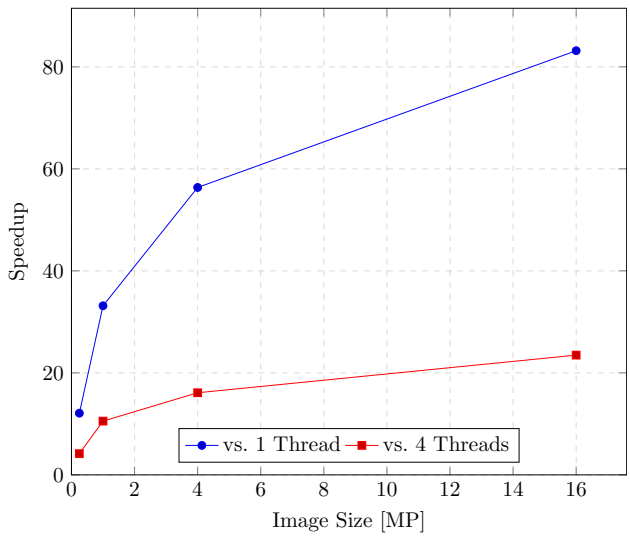


Fig. 7. Speedup obtained on a GTX 770, compared with the single-threaded and 4-threaded CPU implementation running on an Intel Core i7-4770.

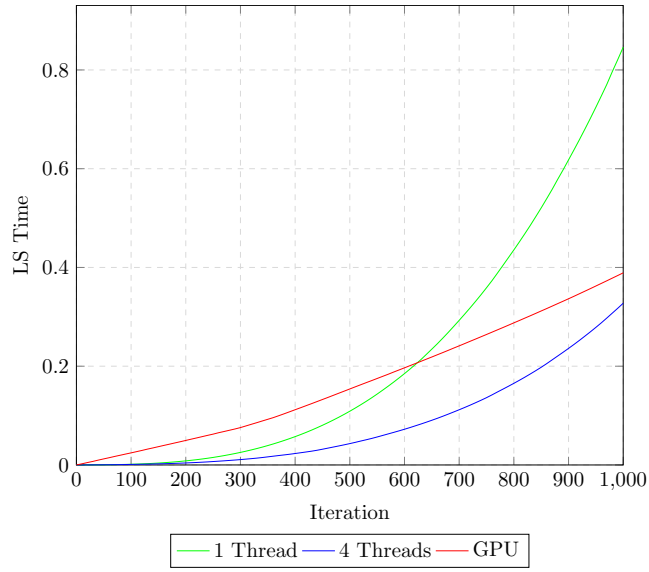


Fig. 8. Time taken to re-weight the selected pixels (matrix inverse update), as a proportion of the total runtime for that implementation.

Figure 8 shows the time spent calculating the inverse matrix. Initially, the inverse matrix is small enough that the update requires almost no work. As such the GPU implementation initially suffers due to the relatively large kernel execution overhead. However, as the workload increases, this overhead becomes less significant, and the GPU implementation exhibits slower growth than the CPU implementation.

Since the time taken to compute the inverse matrix is independent of the image size, its contribution to the total runtime becomes insignificant for larger images. When deconvolving a 16 MP image, this contribution is less than 1% after 1000 iterations.

8. Conclusion

Observing the radio sky with a radio interferometer results in an image of the sky that is convolved with the PSF. Deconvolution algorithms attempt to recover the true sky from this convolved image. Radio astronomers typically use a variant of the CLEAN algorithm for this purpose.

Based on the assumption of a sparse sky model, we instead consider Compressed Sensing techniques for deconvolution. In particular, we adapt the OMP algorithm for deconvolution.

Since a radio interferometer measures points on the 2D Fourier plane, we can adapt OMP to use

FFTs instead of DFT matrices. This allows us to reduce the complexity of OMP, which allows for a feasible runtime for standard image sizes.

In order to evaluate the effectiveness of OMP for deconvolution, we generate synthetic sky models and run them through a simulated interferometer. We can then deconvolve the resulting images with both OMP and CLEAN, and compare the results to the initial sky model.

In terms of image quality, we find that OMP extracts significantly more sources than CLEAN, extracting up to 82% of the sources over the 100 sky models, compared with CLEAN's 61%. In addition, the residual after source extraction is 2.7 times lower for OMP than for CLEAN.

For OMP, the residual is elevated around the edges of the image, resulting in occasional spurious sources in this region. As such, a wider image should be used, which can be cropped after deconvolution. Due to the lower residuals, there are also some low-intensity spurious sources which would be lost in the noise of the CLEAN image.

Since future radio interferometers, the Square Kilometer Array in particular, expect to have significantly larger images to deconvolve. Deconvolution of these larger images thus requires faster deconvolution implementations.

As such, we adapt our OMP implementation to make use of parallel architectures. In particular, we create a multi-core CPU implementation using OpenMP, and a GPU implementation using nVidia's CUDA.

Our GPU implementation achieves an 83× speedup over the single-threaded implementation, and a 23× speedup over the 4-threaded CPU implementation (that is, our parallel CPU implementation achieves a near-linear speedup at 4 CPU cores).

9. Future Work

OMP, like CLEAN, is designed to deconvolve point sources. However, CLEAN can still prove surprisingly effective at resolving extended structures. It would be beneficial to evaluate how OMP performs when tasked with deconvolving an extended emission.

Additionally, there are several variants of CLEAN that increase its effectiveness when deconvolving extended structures transforming the image into a sparser basis, such as multi-scale CLEAN or wavelet CLEAN. In this, a transform of the PSF is subtracted from the transform of the dirty image to produce a transform of the deconvolved image.

A similar approach could prove beneficial to OMP for deconvolving extended structures, however the algorithmic optimizations will need to be reworked using the new sensing matrix. In particular, some of the optimizations might become impossible under some transforms (particularly those which cannot be expressed as a matrix transform), which would result in infeasible execution times.

OMP perfectly optimizes the intensity of all selected sources in each iteration. However, bad positioning of a source might never be corrected throughout the algorithm execution. It might further reduce the residual if an OMP algorithm was developed which also optimized the source position. This might be achieved by also adding several pixels near each source when performing the least-squares optimization. The pixel with the maximum resulting intensity should be the best candidate for a source.

Additionally, if the maximal intensities diverge in two or more directions from the initial source, this could indicate that multiple sources are within the resolving ability of the interferometer, and could be a method to achieve super-resolution. Alternatively, this could be set off by noise, resulting in many more spurious sources and reducing the image sparsity.

The constant improvement of technology, particularly graphics hardware, makes writing future-proof high-performing software a challenge. In order to combat this, our OMP implementation abstracts the specific hardware by using libraries likely to be frequently updated to make use of the latest hardware optimizations. Thus, so long as the library interfaces remain consistent, this implementation should retain good performance on future hardware. As such, this implementation should be tested on newer hardware and larger data sets in the future.

Acknowledgments

The financial assistance of the National Research Foundation (NRF) towards the research is hereby acknowledged. Opinions expressed and conclusions arrived at, are those of the author and are not necessarily to be attributed to the NRF.

References

Candès, E. J. [2006] "Compressive sampling," in *Proc. Int. Congress of Mathematicians: Madrid*, August 22–30, 2006, invited lectures, p. 1433.

Candes, E. & Romberg, J. [2007] *Inverse Prob.* **23**, 969.

Carrillo, R., McEwen, J. & Wiaux, Y. [2012] *MNRAS* **426**, 1223.

Cotton, W., Schwab, F., Perly, R. & Bridle, A. [1989] *Synthesis Imaging in Radio Astronomy* (Astronomical Society of the Pacific), p. 243.

Dagum, L. & Menon, R. [1998] *IEEE Comput. Sci. Eng.* **5**, 46.

Fang, Y., Chen, L., Wu, J. & Huang, B. [2011] "GPU implementation of orthogonal matching pursuit for compressive sensing," in *IEEE 17th Int. Conf. Parallel and Distributed Systems (ICPADS), 2011* (IEEE), p. 1044.

Frigo, M. & Johnson, S. G. [1998] "FFTW: An adaptive software architecture for the FFT," in *Proc. 1998 IEEE Int. Conf. Acoustics, Speech and Signal Processing, 1998* (IEEE), p. 1381.

Garsden, H., Girard, J., Starck, J.-L. *et al.* [2015] *A&A* **575**, A90.

Harris, M. [2013] "Unified memory in cuda 6," URL devblogs.nvidia.com/parallelforall/unified-memory-in-cuda-6.

Hoberock, J. & Bell, N. [2009] "Thrust: C++ template library for cuda," URL code.google.com/p/thrust.

Högbom, J. [1974] *A&ASS* **15**, 417.

Jonas, J. L. [2009] *Proc. IEEE* **97**, 1522.

Lawson, C. L. & Hanson, R. J. [1974] *Solving Least Squares Problems*, Vol. 161 (SIAM, USA).

Li, F., Cornwell, T. J. & de Hoog, F. [2011] arXiv:1106.1711.

Lustig, M., Donoho, D. & Pauly, J. M. [2007] *Magn. Reson. Med.* **58**, 1182.

Mallat, S. G. & Zhang, Z. [1993] *IEEE Trans. Signal Process.* **41**, 3397.

Mohan, N. & Rafferty, D. [2015] *Astrophysics Source Code Library.*

Narayan, R. & Nityananda, R. [1984] "Indirect imaging," *Proc. IAU/URSI Symp.*, ed. Roberts, J. A. (Cambridge University Press, London), p. 267.

Nvidia, C. [2008] "Cublas library," URL developer.nvidia.com/cuBLAS.

Nvidia, C. [2010] "Cufft library," URL developer.nvidia.com/cuFFT.

Pati, Y. C., Rezaiifar, R. & Krishnaprasad, P. S. [1993] "Orthogonal matching pursuit: Recursive function approximation with applications to wavelet decomposition," in *1993 Conf. Record of the Twenty-Seventh Asilomar Conf. Signals, Systems and Computers, 1993* (IEEE), p. 40.

Schwardt, L. C. [2012] "Compressed sensing imaging with the kat-7 array," in *Int. Conf. Electromagnetics in Advanced Applications (ICEAA), 2012* (IEEE), p. 690.

Septimus, A. & Steinberg, R. [2010] "Compressive sampling hardware reconstruction," in *Proc. 2010 IEEE Int. Symp. Circuits and Systems (ISCAS)* (IEEE), p. 3316.

Shaobing, C. & Donoho, D. [1994] "Basis pursuit," in *28th Asilomar Conf. Signals, Systems Computers.*

Starck, J., Pantin, E. & Murtagh, F. [2002] *Publi. Astrono. Soc. Paci.* **114**, 1051.

Torrellas, J., Lam, M. S. & Hennessy, J. L. [1994] *IEEE Trans. Computers* **43**, 651.

Tropp, J., Gilbert, A. C. *et al.* [2007] *IEEE Trans. Inf. Theory* **53**, 4655.

Wang, E., Zhang, Q., Shen, B. *et al.* [2014] "Intel math kernel library," in *High-Performance Computing on the Intel Xeon Phi* (Springer, NY), p. 167.

Wiaux, Y., Jacques, L., Puy, G. *et al.* [2009] *MNRAS* **395**, 1733.