P9 Distributed Image Reconstruction for the new Radio Interferometers

Jonas Schwammberger

October 8, 2019

Abstract

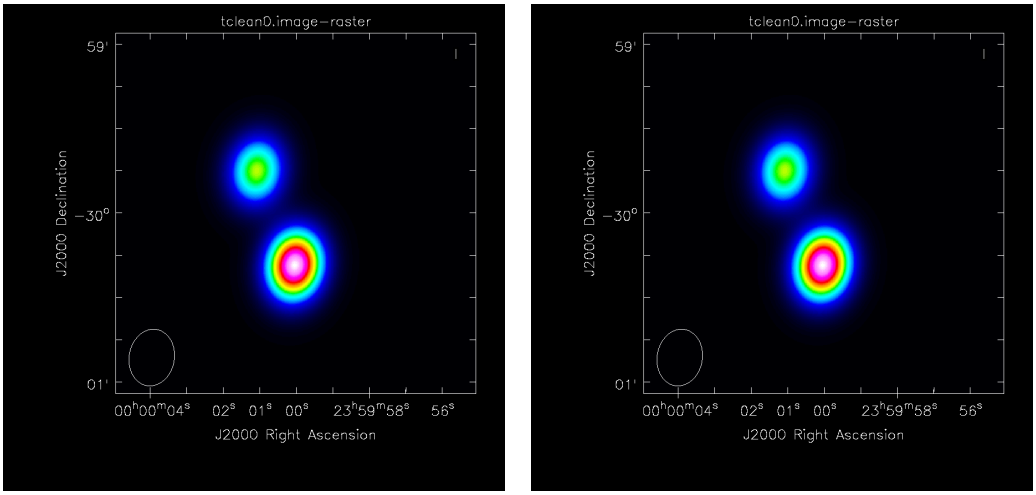
Contents

1 The image reconstruction problem of radio interferometers

In Astronomy, one goal is to find ever smaller objects in the sky. For this purpose, we build instruments with higher angular resolution. The instruments angular resolution depends on two factors: On the diameter of the antenna-dish or mirror, and on the observed wavelength. With longer wavelengths we need bigger dishes/mirrors to achieve a similar angular resolution.

This is an issue for Radio Astronomy. The long radio wavelengths require huge dishes for a high angular resolution. Of course there is a practical limit on the antenna-dish diameter we can build. The famous Arecibo observatory is one of the largest single-dish radio telescopes with a diameter of 305 meters. Antennas with such a large diameter become difficult to steer accurately, let alone the construction costs. We have reached the practical limits of single-dish telescopes. If we require higher angular resolution, we need to look at another type of instrument: The radio interferometer. They use several smaller antennas together, acting like a single large dish. An interferometer can achieve angular resolutions which are comparable to dishes with a diameter of several kilometers.

But there are drawbacks: The interferometer does not measure the sky in pixels. It measures the sky in Fourier space. As such, an interferometer produces amplitude and phase for each Fourier component that was measured. The observed image has to be reconstructed from the Fourier measurement. The measured Fourier components are called visibilities in the Radio Astronomy literature. From this point forward, we will call the measured Fourier components visibilities. The Figure 1 shows an example of the image reconstruction problem. The Figure 1a shows the measurements in the Fourier space, and the figure 1b shows the observed image of the sky, with two stars close to each other. The image reconstruction has to find the observed image 1b from the measurements 1a.



(a) Measurements of the interferometer in the Fourier space.

(b) Observed image of the sky, showing two stars.

Figure 1: The image reconstruction problem, the observed image has to be reconstructed from the Fourier measurements.

At first glance, we might believe that the image reconstruction is trivial: The interferometer measures Fourier components, and efficient algorithms for the inverse Fourier transforms are well-known. However, two properties of the measured Fourier components make the image reconstruction difficult: The measurements are both noisy and incomplete.

The atmosphere of the earth is one source that introduces noise. It adds noise to the amplitude and phase of each measured Fourier component. The atmosphere changes over time and can under the right circumstances introduce a high level of noise compared to the signal. The image reconstruction should be able to

find the observed image from potentially very noisy Fourier measurements.

The interferometer measures an incomplete set of Fourier components. Note that the Figure 1a shows the Fourier space, which has missing components. The interferometer can only measure a limited set of Fourier components. The reconstruction algorithm has to find the observed image even though important Fourier components are missing from the measurements.

These two difficulties, the noise and the incomplete measurements, lead to the fact that there are many different candidate images that fit the measurements. This is known as an inverse problem. We want to find the observed image, even though all we have are imperfect measurements. From the measurements alone, we cannot decide which candidate is the truly observed image. However, we have additional knowledge that simplifies the inverse problem: We know it is an image of the sky, which consists of stars, hydrogen clouds, etc. By including prior knowledge in the reconstruction, we can find the most likely image given the measurements.

The question remains is: How close is the most likely image to the observed one? Is exact reconstruction possible where the most likely and observed image are equal? Surprisingly the answer is yes. It is possible in theory[1, 2], and was shown in practice on low noise measurements[3, 4]. However, not all algorithms perform equally well when the noise level in the measurements is high. Also, computing resources required for each algorithm can vary significantly. In short, a reconstruction algorithm has three opposing goals:

1. Produce a reconstruction which is as close to the truly observed image as possible.
2. Robust against even heavy noise in the measurements.
3. Use as few computing resources as possible.

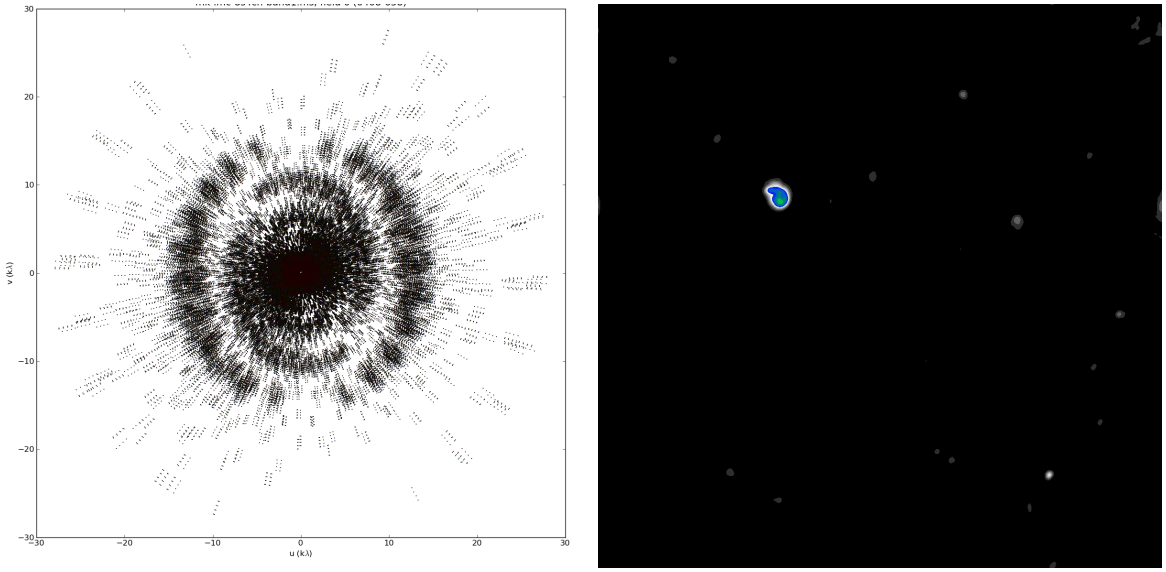
No reconstruction algorithm performs equally well on all three goals. One of the most widely used reconstruction algorithms is CLEAN [5, 6]. It has shown to be robust against heavy noise and, depending on the observation and is one of the oldest algorithms still in use today. As such, it was developed before the advent of distributed and GPU-accelerated computing. Today's new radio interferometers produce ever more measurements. The recently finished MeerKAT radio interferometer produces roughly 80 million Fourier measurements each second. Astronomers wish to reconstruct an image from several hours worth of measurement data. Reconstructing an image from this data volume requires GPU and distributed computation. But how to use GPU and distributed computing effectively is still an open problem.

Coordinate descent methods have been successfully applied in other inverse problems, such as reconstruction of CT scans[7], or X-Ray imaging[8]. GPU accelerated[9] and distributed[10] variants have been developed. To our knowledge, coordinate descent methods have not been explored for the inverse problem in Radio Astronomy.

In this work, we develop our own proof-of-concept image reconstruction algorithm based on coordinate descent methods. We apply the reconstruction on a real world MeerKAT observation provided by SARAO. We explore the possible speedups we can achieve by using GPU and distributed computation. The algorithm is implemented platform independent in .netcore.

The rest of this work is structured as follows. First in section 2, we give an introduction to radio interferometric imaging, and give the theoretical background to why a reconstruction can even achieve a higher resolution than the instrument. Next we present the current state-of-the-art in image reconstruction for radio interferometers in section 3. Then we derive a basic image reconstruction algorithm based on coordinate descent in section 4, and show how we can use GPU acceleration and distribution to speed up reconstruction.

2 Introduction to radio interferometric imaging



(a) Sampling Pattern in Fourier space. *(b) Reconstructed image.*

Figure 2: Example of an image reconstruction for Fourier measurements of the MeerKAT radio interferometer

Radio interferometers sample an incomplete set of Fourier components of the sky. Each Fourier component consists of the noisy amplitude plus phase measurements and the uv location on where it was measured in the Fourier space. Figure 2a shows the location of the measurement points in Fourier space of a real-world MeerKAT observation. We wish to reconstruct the observed image 2b from the measurements.

T

From the measurements alone, this is not possible. We need to add prior knowledge about the image. Ill-posed inverse problem It is an optimization problem. We need to find the optimum between being as close to the measurements, and being consistent with our prior information.

Radio astronomy is difficult, a lot more effects than we have time to handle. Fourier components are called visibilities in

2.1 From electromagnetic waves over visibilities to images

We give a short introduction into how the electromagnetic wave gets measured by the interferometer, turned into visibilities and finally processed into an image. Figure 3 shows a radio source and its electro-magnetic (em) wave arriving at the antennas of the interferometer. It then shows the three processes involved to arrive at an image: Correlation, calibration and image reconstruction.

First, we have a source in the sky that is emitting em-waves in the radio frequency. The waves travel to earth, through the earth's ionosphere and finally to our interferometer. Along its path, the e-m waves may get distorted from various sources. For example, it may receive a phase shift by the ionosphere.

Then, the em-wave arrives at our interferometer. We call each antenna pair a baseline. Each baseline will end up measuring a single visibility. The distance between the antennas and their orientation to the em-wave will determine where we sample the uv-plane. Short baselines measure the uv-plane close at the origin, while long baselines sample the uv-space further away from the origin. Remember that the samples away from the uv-origin contain the information about edges and other details of our image. With a longer baseline the

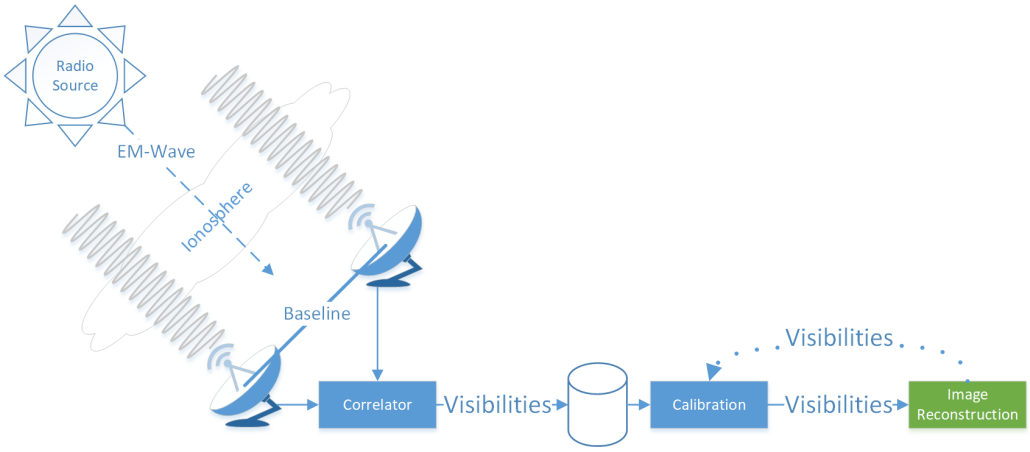


Figure 3: Radio interferometer system

interferometer measures more highly resolved details, regardless of the antenna dish-diameter[1]. The figure 3 shows the em-wave arriving at a single baseline of the interferometer. Each antenna picks up its version of the em-wave and transfers it to the correlator.

The correlator then takes the feed of each antenna and correlates the signals, which results in the amplitude and phase of the visibility component. Amplitude and phase for each visibility are measured for a short time range (i.e. fractions of a second up to several seconds). At this point, the visibilities are saved to disk for further processing. The radio interferometer produces a visibility measurement for each baseline, for each time range, for each frequency channel of the instrument. Because a single observation can take up to several hours, measured with several thousand frequency channels, radio interferometers produce an almost arbitrary large number of visibilities.

Calibration

Image Reconstruction

2.1.1 The measurement equation

As we discussed so far, the radio interferometer measures visibilities of the sky image, and we wish to find the observed image from the measurements. Put formally, we wish to invert the following system of linear equations (2.1), where V is the visibility vector[2], F is the Fourier transform matrix and I is the pixel vector of the observed image.

$$V = FI \tag{2.1}$$

We wish to find the observed image I, while we only know the visibility vector V and the Fourier transform matrix F. This is what we call the measurement equation. In most context for this project, looking is an adequate view of the image reconstruction problem. We will show why we cannot find the observed image I by simply calculating the inverse Fourier transform. However, when we need to efficiently apply the Fourier transform, we need to know F in more detail. As we will see, radio interferometers have some difficulties hidden in the Fourier transform matrix, which are difficult to handle efficiently. First, let us abandon the vector

[1] Remember that this is the reason why we build radio interferometers. We do not need impossibly large dish diameters for a high angular resolution. We just need large distances between smaller antennas.

[2] We use the lower-case v to denote the axis in the Fourier space uvw, and the upper-case letter to denote the visibility vector.

notation of (2.1), and represent the measurement equation with integrals (2.2).

$$V(u,v) = \int\int I(l,m)e^{2\pi i[ul+vm]}\,dl\,dm \tag{2.2}$$

This is essentially the same problem. The main difference is that we do not represent the Fourier transform as a matrix F, but as integrals $\int\int e^{2\pi i[ul+vm]}$, where u, v are the coordinates in Fourier space and l, m are the angles away from the image center. A single pixel represents the intensity of the radio emission from the direction l, m. Note that the measurement equation (2.2) shows the fact that the visibilities are measured in a continuous Fourier space. If the Fourier space would also be discrete, we could replace the integrals with sums.

However, the measurement equation (2.2) is inaccurate in the sense that it ignores many effects that distort the signal. For example, it does not account for the distortion by the ionosphere, or the distortion introduced by real-world antennas. The measurement equation (2.2) shown here does not represent the real world. But depending on the instrument and the observation, these distortions may be negligible, and the measurement equation (2.2) is a good approximation.

When there is a distortion source that cannot be ignored, it has to be modelled in the measurement equation. As such there is no unified measurement equation for all radio interferometric observations, let alone radio interferometers. The equation shown in (2.2) can be seen as the basis that gets extended as necessary[11, 12, 13, 14].

For example, the measurement equation (2.2) is only accurate for small field of view observations, when l and m are both small angles. For wide field of view observations, we need to account for the fact that the visibilities have a third term w, and we arrive at the wide field of view measurement equation (2.3).

$$V(u,v,w) = \int\int \frac{I(l,m)}{c(l,m)} e^{2\pi i[ul+vm+w(c(x,y)-1)]}\,dl\,dm\,, \quad c(l,m) = \sqrt{1-l^2-m^2} \tag{2.3}$$

The third w-term has two effects on the measurement equation. It introduces a phase shift in the Fourier transform $e^{2\pi i[\ldots+w(c(l,m)-1)]}$, and a normalization factor of the image $\frac{I(l,m)}{c(l,m)}$. Note that when the angles are small, i.e. $l^2 + m^2 \ll 1$ then the wide field of view measurement equation (2.3) reduces to our original (2.2). This is another way of saying that for small field of views, the measurement equation (2.2) is a good approximation under the right conditions.

In this project, we use the wide field of view measurement equation (2.3). But as we mentioned in the beginning of this section, for most contexts, it is not important whether we ignore the w-term of the visibilities or not. It is important when we design an efficient implementations for applying the wide field of view Fourier transform, because the w-term keeps us from using the Fast Fourier Transform (FFT). In every other case, we can ignore this technicality. Because even more complicated measurement equation still have a linear relationship between visibilities and image [11, 12, 13, 14]. We can view the whole reconstruction problem as a system of linear equations (2.1), where the matrix F takes care of how exactly the measurements and pixels relate in this case.

2.2 The ill-posed image reconstruction problem

So far, we discussed how the interferometer measures in Fourier space, and we wish to find the observed image that matches the measurements. In other words, we wish to find a solution to a system of linear equation (2.4), where V are the measurements, x is the observed image and F is the Fourier transform matrix. We also discussed that F can be complicated in practice, but is still essentially a linear operator. Meaning we know how the inverse Fourier transform matrix F^{-1}, and the question arises: Why can't we solve

the equations (2.4) by calculating the inverse Fourier transform? Or, why does $x = F^{-1}V$ not lead to the observed image?

$$\underset{x}{solve} \quad V - Fx = 0 \tag{2.4}$$

The answer is, the equations (2.4) do not have a unique solution, which makes the problem ill-posed. The problem is considered ill-posed when it has one of the following properties:

1. No solution exists.
2. There are solutions, but no unique solution exists.
3. The solution behaviour does not change continuously with the initial condition (For example: a small change in the measurements lead to a very different reconstructed image).

From linear algebra, we know that an under-determined system of linear equations, i.e. when (2.4) has more pixels than visibility measurements, then the problem is under-determined and there may be a potentially infinite number of solutions to the system. Under-determined systems arise in many similar fields, as for example in X-Ray imaging of the sun[8]. However, radio interferometers measure a large number of visibilities. We generally have more visibilities than pixels. This means the image reconstruction problem (2.4) for radio interferometers is actually over-determined.

From linear algebra, we know that an over-determined system either has one or zero solutions. At first glance it may be counter-intuitive why there are many possible solutions to (2.4) for radio interferometers. The reason why lies in two properties of the measurements: They are noisy and incomplete.

The radio interferometer measures noisy visibilities, meaning each amplitude and phase of a measurement is influenced by an unknown noise factor. Finding a reconstructed image is the same as finding the de-noised versions of the visibility measurements. This alone would make the problem ill-posed, but the visibilities actually have a second property that makes them ill-posed: incompleteness.

When we look back at figure 2a, it is clear to see that the interferometer does not sample the visibilities in a uniform way. There are regions with a high sample density. The density decreases when we move further away from the center. The higher frequency visibilities get fewer and fewer samples. This means we are missing data for crucial measurements for the reconstruction.

Also note that the question whether the measurements are incomplete essentially comes down to the image resolution of the reconstruction: Since we are missing high-frequency measurements and we can choose the resolution of the image, we can also reduce the resolution of the reconstructed image until the missing frequencies become negligible. However, if we can solve the ill-posed inverse problem, we can reconstruct an image at a higher resolution from the same measurements. The question is how do we solve the ill-posed inverse problem?

2.2.1 Adding a regularization

For ill-posed inverse problems, there are two viewpoints for the same idea. From the viewpoint of optimization, we can solve the ill-posed image reconstruction problem (**??**) by adding a regularization. The regularization creates a system of linear equations with a unique solution. From the viewpoint of Bayesian Statistics, we include prior knowledge about the image, and therefore search the most likely image given the measurements. For this project, both terms describe the same idea and we use regularization and prior interchangeably. We know that the reconstructed image from radio interferometers contain a mixture of stars (point sources located in a single pixel) and extended emissions like hydrogen clouds. By adding this prior knowledge to the reconstruction problem, we can find the most likely image given the measurements. As we will see, under the

right prior, we can create a reconstruction algorithm that is almost guaranteed to find the truly observed image in theory.

There are different ways to include regularization in the reconstruction problem. In this project, we use the following method: We cast the image reconstruction problem into an optimization objective consisting of a data fidelity term and a regularization term. A reconstruction algorithm therefore consists of an optimization objective, a prior function and an optimization algorithm.

$$\underset{x}{minimize}\ \|V - Fx\|_2^2 + \lambda P(x) \tag{2.5}$$

The objective (2.5) has a data term $\|V - Fx\|_2^2$, which forces the most likely image to be as close to the measurements as possible, and the regularization term $P(x)$, which penalizes unlikely images according to our prior function. The parameter λ represents how much we penalize unlikely images and by extend, much noise we expect in the reconstruction. The parameter λ is either left to the user to define, can be estimated from the data [15].

The prior function $P()$ represents our prior knowledge about the image. It assigns a high penalty for unlikely images. In in radio interferometric image reconstructions tend to use similar prior functions as for image denoising applications. Such as: Total Variation ($\|\nabla x\|_1$) [16], L2 ($\|x\|_2$) [17] or the L1 norm in a wavelet space ($\|\Psi x\|_1$) [18].

Finally, an optimization algorithm is necessary which can optimize the objective function (2.5) with the chosen prior function $P()$. Typical choices for optimization algorithms in image reconstructions are Interior Point Methods like Simplex, Matching Pursuit [5] and ADMM[19].

A reconstruction algorithm for radio astronomy consists of an optimization objective, a prior function and an optimization algorithm. It is a three dimensional design space. Not every prior is suitable for every optimization algorithm. The choice of optimization objective influences both what prior and what optimization algorithm we can use. Although there are a different choices for the optimization objective, we limit ourselves to the objective (2.5) and explore how we can distribute the reconstruction problem.

The last question that remains is, how close are the most likely image, under a given prior, to the truly observed one? Remember that the Nyquist-Shannon sampling theorem states that our uniform sampling frequency needs to be larger than twice the highest frequency in a band-limited signal[3], and then the theorem guarantees exact reconstruction. For radio astronomy, we do not have uniformly sampled visibilities, and although we have a large number of samples, we are missing crucial parts of the Fourier space. Luckily, there is another sampling theorem that, under certain assumptions, guarantees exact reconstruction for the case of radio interferometers: The theory of compressed sensing.

2.2.2 Compressive sampling of the sky

For the sake of demonstration, let us assume the radio interferometer observes a patch of sky containing ten stars. It measures an incomplete set of random Fourier components of the ten stars, and we would like to reconstruct an image of size 256^2 pixels. The emissions from stars are concentrated into a single pixel. For compressive sampling, we need to know an space in which our reconstructed image is sparse, and we need to take measurements in a different, incoherent space.

Our reconstructed image contains only zero pixels except at the ten locations of the stars, meaning the image space for this patch of the sky is already sparse. In this case, we do not need any additional sparse space

[3] For example: if we want to record human voices with the highest frequency of 20 kHz, Nyquist-Shannon states our uniform sampling frequency has to be larger than 40 kHz to guarantee exact reconstruction

like wavelets. We can reconstruct directly in the sparse image space, and we have the first requirement met for compressive sampling.

The next requirement is that the measurement and reconstruction space (which is the image in this example), are as incoherent as possible.

Yes, incoherence.

$$\underset{x}{minimize} \ \|V - Fx\|_2^2 + \lambda \|x\|_1 \tag{2.6}$$

Intuitively, the number of samples depend on the information content, not on the bandwidth. thi

At what point are we guaranteed? The matrix A needs to fullfil the Restricted Isometry Porperty (RIP) [1, 2]. Approximately orthonormal on sparse signals. (If we randomly choose ten columns of F, how much do they correlate. We want as little correlation as possible.) Calculate the RIP is NP-hard[20]. Approximations are also difficult to compute[21]. So we can only talk about how likely a given matrix fullfils the RIP. Matrix where each element is sampled from a random Gaussian distributîon has the highest change.

Random samples in the Fourier space also have a high chance to fulfill the RIP [22].

Not possible for Radio interferometers, because they sample in the Fourier space. Not random.

There are also extended emissions, clouds etc. Resulting in a lot of non-zero pixels. There may be better sparse spaces for radio astronomy.

But also RIP may be too strict. Exact reconstruction can also work under less strict conditions[23] active field of research.

2.2.3 Reconstruction guarantees in the real world

What does this all mean for image reconstruction? For us, we design reconstruction algorithms and cannot influence the matrix F. We do not know if it actually

The theory of compressed sensing gives us a framework. There is a data modelling task, and a task for finding efficient algorithms.

So there is a data modelling task in finding a good sparse prior. We also do not know the proper sparse space in which radio interferometric images. We know several spaces, Curvelets [24] Starlets [25], Daubechies wavelets [26]. As of the time of writing, it is currently unknown which leads to the best reconstruction. We can also learn dictionaries.

The task for efficient algorithms is finding the best way to optimize the objective with a given prior on real hardware. Fast convergence. Also difficult, because in practice the choice of prior can also influence convergence.

Compressed sensing based reconstruction algorithm. In radio astronomy.

2.3 Introduction into optimization/RI reconstruction algorithms

2.3.1 The Major/Minor cycle

2.3.2 Image reconstruction as deconvolution

3 State of the Art image reconstruction

Image reconstruction pipelines are split in two taks. Gridding and deconvolution. We introduce here the two latest gridding algorithms, w-stacking in 3.1 and Image Domain Gridder 3.2.

Deconvolution we discuss CLEAN and MORESANE.

Not all algorithms are here from

3.1 w-stacking Gridder

3.2 Image Domain Gridding Algorithm

3.3 Deconvolution

3.3.1 CLEAN

Various Improvements and speedups[6, 27, 28, 29], but the core algorithm is still this.

3.3.2 MORESANE

3.4 Coordinate Descent

4 Coordinate descent deconvolution

Coordinate descent methods are a family of algorithms. Various variants exist[30, 31], but they share one common idea: Most of our problems become simple when we reduce the number of dimensions. Deconvolving a whole image is difficult. But deconvolving a single pixel is easy. As we will show in section 4.2, we can derive a closed form solution[4] for deconvolving a single pixel. We then iterate over all pixels, possibly several times, until we converge to a deconvolved image.

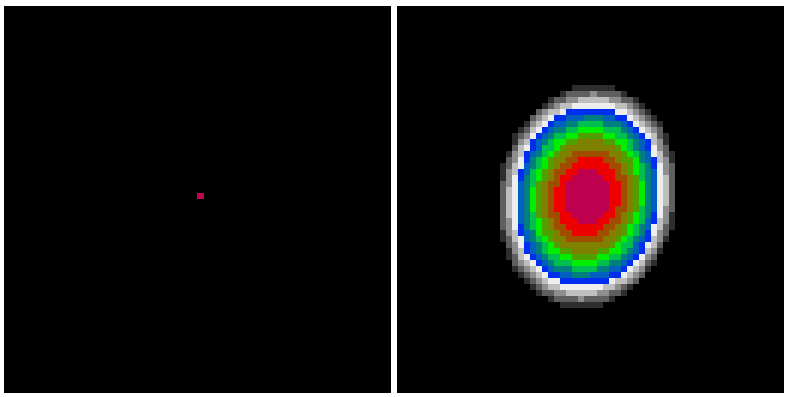
This is the idea behind coordinate descent methods. By reducing the dimensions of the problem, we can often find an optimization algorithm where each iteration is ”cheap” to compute. In these cases, coordinate descent methods produce competitive results[32, 33].

For a deconvolution algorithm in radio astronomy, we need three parts: An optimization algorithm, a regularization, and an optimization objective. We use coordinate Descent as the optimization algorithm, take Elastic Net as the regularization and use the following objective function:

$$\underset{x}{minimize}\ \frac{1}{2}\left\|I_{dirty} - x * PSF\right\|_2^2 + \lambda ElasticNet(x) \tag{4.1}$$

As we have shown before, the objective function consists of two parts. The data term $\|I_{dirty} - x * PSF\|_2^2$ and the regularization term $ElasticNet(x)$. The data term forces the image to be as close to the measurements as possible which forces the image to be as close to the measurements as possible, the regularization term forces the image to be as consistent as possible with our prior knowledge. The parameter λ is a weight that either forces more or less regularization. It is left to the user to define for each image. We will derive the coordinate descent algorithm that optimizes the objective (4.1) in section 4.2. First, let us explain what the elastic net regularization does.

4.1 Elastic net regularization



(a) Effect of the pure L1 norm (λ = 1.0) on a single point source.

(b) Effect of the pure L2 norm (λ = 1.0) on a single point source.

Figure 4: Effect of the L1 and L2 Norm separately.

This regularization is a mixture between the L1 and L2 regularization. The Figure 4 shows the effect of the L1 and L2 norm on a single star. The L1 regularization forces the image to contain few non-zero pixels as possible. It encodes our prior knowledge that the image will contain stars. The L2 regularization on the other hand ”spreads” the single star across multiple pixels. This forces the image to represent extended emissions, like hydrogen cloud, with a large number of non-zero pixels (the L1 norm tends to break extended emissions

[4] Deriving a formula which we can implement in a few lines of code.

apart, only using a handful of non-zero pixels). The L2 norm was already used in other image reconstruction algorithms in radio astronomy[17], with the downside that the resulting image will not be sparse.

Elastic net mixes these two norms together, becoming ”sparsifying L2 norm”. It retains the sparsifying property of the L1 norm, while also keeping extended emissions in the image. Formally, elastic net regularization is defined as the following:

$$ElasticNet(x, \alpha) = \alpha \|x\|_1 + \frac{1-\alpha}{2} \|x\|_2 \tag{4.2}$$

Elastic net has three properties which make it an interesting regularization for coordinate descent: It was shown to speed up convergence rates compared to the pure L1 or L2 norm[34], is a separable function, and has a closed form solution. The first property was not further investigated in this work. The second property, separability, means that we can calculate the regularization for each pixel independently of each other, and we still arrive at the same result. Lastly, we can find a simple formula for each pixel that applies the elastic net regularization:

$$ElasticNetClosedForm(x, \lambda, \alpha) = \frac{max(x - \lambda * \alpha, 0)}{1 + \lambda(1-\alpha)} \tag{4.3}$$

The closed form solution (4.3) of the elastic net regularization is also a mixture of the closed form solutions of the L1 and L2 norm. The closed form solution of the L1 norm is shrinkage: $max(x - \lambda, 0)$, we reduce the pixel value by λ and clamp negative values. For the L2 norm, we divide the pixel value: $\frac{x}{1+\lambda}$.

Note that the shrink operation in this project always clamps negative pixels to zero. We constrain the image to only contain zero or positive pixel values. This has become a widely used constraint in radio interferometric image reconstruction and may lead to improved image quality[35].

Elastic net is the regularization we use throughout this work. It is separable (we can calculate it for each pixel independently) and has an easy to compute closed form solution.

4.2 Deriving the basic coordinate descent deconvolution algorithm

In this section we derive the basic coordinate descent deconvolution algorithm, which minimizes the objective (4.1). Coordinate descent methods have a tendency to need a more iterations to converge compared to other methods like gradient descent. However, when a single iteration is cheap to compute, they can be faster in practice[36]. The elastic net regularization has an easy to compute closed form solution (4.3), and a single iteration is cheap to compute.

We call the coordinate descent algorithm described here ”basic”. Other coordinate descent algorithms in the literature[30, 31, 37] can be seen as generalizations of the ”basic” algorithm. The basic algorithm optimizes a single pixel at each iteration, while other algorithms can optimize one or several pixels.

In this section, we derive the basic coordinate descent algorithm that optimizes a single pixel in each iteration, and iterates over all pixels several times, with a specific strategy, until convergence. There are three types of iteration strategy we can choose:

1. Random: where we choose a pixel to optimize uniformly at random.
2. Greedy: where we first choose the pixel which minimizes our objective the most
3. Cyclic: where we choose a subset of pixels and cycle through them until convergence.

The iteration strategy is not important for convergence. We can for example create a mixture of the different strategies and the algorithm would still converge to the optimum. However, the strategy we choose has an

impact on how many iterations we need until convergence. For example: if the image consists of a single star in the center of the image, a greedy strategy would first optimize the pixel at the center, while a random strategy may waste the computing resources in checking every other pixel several times before finally landing on the center. In this implementation, we chose the greedy strategy. Each iteration takes the best possible step towards the optimum. We arrive at the following coordinate descent algorithm in pseudo code:

```
dirty = IFFT(Gridding(visibilities))
residuals = dirty

x = new Array
objectiveValue = SUM(residuals * residuals) + P(x)
oldObjectiveValue = objectiveValue

do
{
  oldObjectiveValue = objectiveValue

  //the core of the algorithm
  pixelLocation = GreedyStrategy(residuals)
  oldValue = x[pixelLocation]
  optimalValue = oldValue + Minimize(residuals, psf, pixelLocation)
  optimalValue = ApplyElasticNet(optimalValue, lambda, alpha)

  //housekeeping
  x[pixelLocation] = optimalValue
  residuals = (dirty - Convolve(x, psf))
  objectiveValue = 0.5 * SUM(residuals * residuals) + lambda * ElasticNet(x, alpha)
} while (oldObjectiveValue - objectiveValue) < epsilon
```

The core of the algorithm consists of the three functions: $GreedyStrategy()$, $Minimize()$ and $ApplyElasticNet()$. The function $GreedyStrategy()$ will be discussed in Section 4.3. The function $ApplyElasticNet()$ was already described in equation (4.3). The $Minimize()$ function is responsible for minimizing the data term of our objective (4.1). Because we only minimize a single pixel, we are dealing with a one dimensional minimization problem and can derive a closed form solution for it.

When we only have one pixel to minimize, the data term of our objective (4.1) reduces itself to a parabola. We derive the standard parabola form in (4.4), where $\langle x, y \rangle$ is the inner product(element-wise multiplication followed by a sum over all elements):

$$\begin{aligned} Minimize(pixel) &= \|I_{res} - PSF * pixel\|_2^2 \\ Minimize(pixel) &= (I_{res} - PSF * pixel)^2 \\ Minimize(pixel) &= \langle I_{res}, I_{res} \rangle - 2\langle I_{res}, PSF \rangle * pixel + \langle PSF, PSF \rangle * pixel^2 \\ Minimize(pixel) &= \langle PSF, PSF \rangle * pixel^2 - 2\langle I_{res}, PSF \rangle * pixel + \langle I_{res}, I_{res} \rangle \end{aligned} \tag{4.4}$$

Now finding the optimal value for the pixel is the same as finding the optimal value of the parabola:

$$\begin{aligned} f(x) &= a * x^2 \\ &+ b * x \\ &+ c \end{aligned} \qquad \begin{aligned} Minimize(pixel) &= \langle PSF, PSF \rangle * pixel^2 \\ &- 2\langle I_{res}, PSF \rangle * pixel \\ &+ \langle I_{res}, I_{res} \rangle \end{aligned} \tag{4.5}$$

$$x_{min} = \frac{-b}{2a} \qquad pixel_{min} = \frac{-(-2\langle I_{res}, PSF \rangle)}{2\langle PSF, PSF \rangle}$$

This means we can find the optimum value of a single pixel by following the formula in (4.5). Note that the PSF in formula (4.4) and (4.5) is shifted to the pixel position we wish to optimize.

We derived the closed form solution (4.5) by looking at the data objective as a parabola. When we take another look at the closed form solution with Calculus in mind, we can see that the numerator $-2\langle I_{res}, PSF \rangle)$ is actually the same as calculating the gradient for this pixel, and the denominator $\langle PSF, PSF \rangle$ is the Lipschitz constant.

Intuitively, the Lipschitz constant describes how fast a function $f(x)$ changes with x. If $f(x)$ changes slowly, we can descend larger distances along the gradient without the fear for de-convergence. In short, it is a data-defined step size. Because our function $Minimize()$ is simply a parabola, the gradient together with the Lipschitz constant point to the optimum of our function.

Because the minimizer (4.5) of our coordinate descent algorithm calculates the gradient, one might ask what the differentiates the coordinate descent method from gradient descent. The main difference is that coordinate descent uses the gradient of a single pixel (or subset of pixels in other versions), in each iteration, while gradient descent generally uses the gradients of all pixels in each iteration. Conceptually, coordinate descent does is not bound to use the gradient. We could also minimize the pixel with a line-search algorithm, trying different values for the pixel, and it is still a coordinate descent method.

This is how the basic coordinate descent deconvolution algorithm works. But as it is described here, one iteration is too expensive to be practical. The $Minimize()$ function calculates both the gradient and the Lipschitz constant in each iteration and the residual update in line 20 requires a convolution. We can drastically improve the runtime costs by caching intermediate results, and using approximations.

4.3 Efficient implementation of basic coordinate descent deconvolution

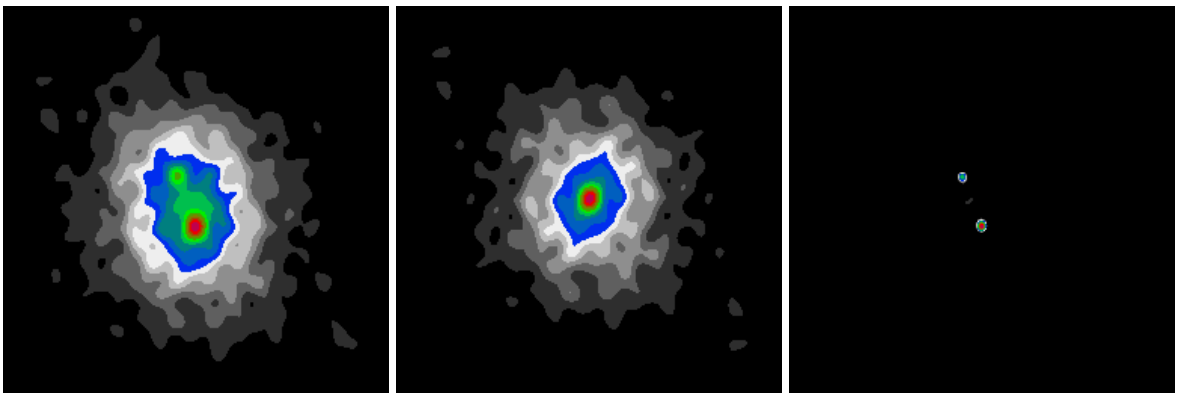
In Section 4.2, we derived the basic coordinate descent deconvolution algorithm. There are several ”tricks” to speed up each iteration. We can cache intermediate results, and exploit the convolution to efficiently calculate the inner products of the basic algorithm We discuss:

1. Edge handling of the convolution
2. Pre-calculation of the Lipschitz constants
3. Efficient greedy strategy
4. Pre-calculation of gradients
5. Efficient update of gradients

Gradient calculation is the most time consuming step. We can exploit the convolution to efficiently pre-calculate, update and approximate the gradients for each pixel. This will be discussed in detail in this Section. To our knowledge, we are the first to explore ways to approximate the gradient in radio interferometric image

reconstruction, and their effect on parallel and distributed deconvolution. As we will se in the later sections, approximating the gradients can help us to distribute the deconvolution.

Visual aid:



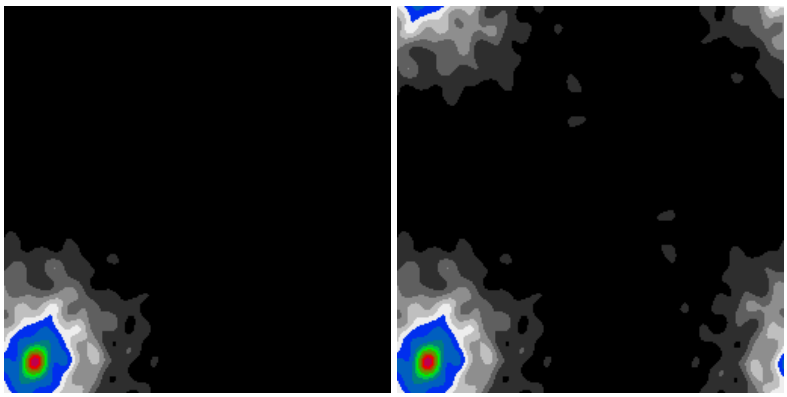
(a) Dirty Image. *(b) Point Spread Function.* *(c) ElasticNet deconvolution*

Figure 5: Example problem with two point sources.

First, we dive into the implementation of the convolution operator and the pre-calculation of the Lipschitz constants, and then we discuss the gradient calculation in detail.

4.3.1 Edge handling of the convolution

As the reader is probably aware, there are several ways to define the convolution in image processing, depending on how we handle the edges on the image. Two possibilities are relevant for radio interferometric image reconstruction: Circular and zero padded.



(a) Zero padded convolution. *(b) Circular convolution.*

Figure 6: Comparison of the two convolution schemes.

Circular convolution assumes the image ”wraps” around itself. If we travel over the right edge of the image, we arrive at the left edge. The convolution in Fourier space is circular. Remember: A convonlution in image space is a multiplication in Fourier space, and vice versa. When we convolve the reconstructed image x with the PSF using circular convolution, then non-zero pixels at the right edge of the image ”shine” over to the left edge. This is physically impossible.

Zero padding assumes that after the edge, the image is zero. Non-zero pixels at the right edges of the image do not influence the left edge after convolution. This is the physically plausible solution. However, the zero

padded convolution is more expensive to calculate. We either have to calculate the convolution in image space, which is too expensive for large kernels, or apply the FFT on a zero-padded image. Either way, it is more expensive than the circular convolution.

In designing a deconvolution algorithm, we have the choice between the circular and the zero-padded convolution scheme. Circular convolution is more efficient to calculate, while zero-padded convolution is closer to the reality. Both choices are possible. The PyMORESANE reconstruction algorithm [38] leaves this choice to the user. We decided on using the zero-padded convolution. This choice influences other parts of the coordinate descent deconvolution algorithm, like how we can efficiently calculate the Lipschitz constants.

4.3.2 Pre-calculation of the Lipschitz constants

Lipschitz constants are $\langle PSF, PSF \rangle$. We simply multiply the PSF with itself and sum up the values. However, we are using the zero-padded convolution. This means the PSF for pixels at the edges is not only shifted, but also cropped. In other words, every pixel has a different Lipschitz constant depending on how much the PSF gets cropped by the image edges.

Note that it is not an issue for convergence: The Lipschitz constant describes the largest step we can take without overshooting the target. We can always make smaller steps, but may pay it with more iterations. The Lipschitz constant of the edge pixels is always lower than the center. The coordinate descent algorithm does converge, but needs more iterations for pixels at the edges of the image. Luckily, there is a way to re-use intermediate results, and efficiently calculate the Lipschitz constant for each pixel in the image.

The first observation is that the image edges always create a rectangular crop of the PSF. To calculate the Lipschitz constant, we square and sum up all the values that lie inside the rectangle. This can be exploited with a scan algorithm: We store the PSF as a running sum of squares.

```
var scan = new double[,];
for (i in (0, PSF.Length(0))
{
  for (j in (0, PSF.Length(1))
  {
    var iBefore = scan[i - 1, j];
    var jBefore = scan[i, j - 1];
    var ijBefore = scan[i - 1, j - 1];
    var current = PSF[i, j] * PSF[i, j];
    scan[i, j] = current + iBefore + jBefore - ijBefore;
  }
}
```

$scan[0, 13]$ contains the sum of the squared PSF values from index $(0, 0)$ up to and including index $(0, 13)$. The last element, $scan[PSF.Length(0) - 1, PSF.Length(1) - 1]$ contains the sum of squares over the whole PSF. In short, we have stored the sum of all possible rectangles starting from index $(0, 0)$. If a part of the PSF is cropped, we can look up $scan$ and find out by how much it affects the total sum.

Up to 4

4.3.3 Efficient Greedy strategy

Calculate what each update would lead to what objective. Expensive to calculate.

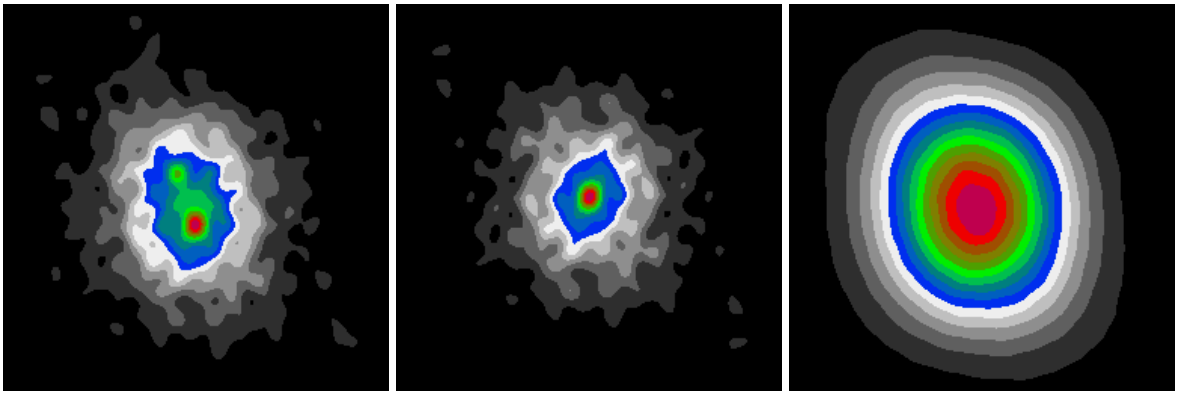
However, we can use another strategy, we use the biggest change in pixel value. A lot cheaper to compute Biggest change in pixel value is in toy examples the same as the best pixel.

Not sure if this is always true.

4.3.4 Pre-calculation of gradients

In each iteration, we need to know the gradients of all pixels. We need to calculate the inner product $\langle I_{res}, PSF \rangle$ for each pixel. It is easy to see that this becomes expensive. When the PSF and the image have the same number of pixels, we need a quadratic number of operations to calculate all gradients.

But this is not necessary. The inner product $\langle I_{res}, PSF \rangle$ is equivalent to calculating the correlation of the residuals with the PSF. This can be done efficiently in Fourier space. Remember that the convolution and correlation operators are related: The convolution is equal to a correlation with a flipped kernel.



(a) Dirty Image. *(b) Point Spread Function.* *(c) Gradient for each pixel.*

Figure 7: Example of the gradient calculation.

To calculate all gradients, we first flip the PSF then calculate the FFT. Then we can proceed the same way as calculating a convolution in Fourier space:

```
1 var psfCorrelationFourier = FFT(FLIP((PSF))
2 var residualFourier = FFT(residuals)
3 var gradients = iFFT(residualFourier * psfCorrelationFourier)
```

We can efficiently calculate all gradients by performing the correlation in Fourier space.

Use the FFT. Pad and flip the kernel. Now we can have all the gradients for each pixel saved to a lookup-map. How we can efficiently update the lookup map in each iteration.

It is faster to calculate the gradient lookup map than $\langle I_{res}, PSF \rangle$ for each pixel.

4.3.5 Efficient update of gradients

Map of gradients. How can we efficiently update this map after we calculated the deconvolution for a single pixel.

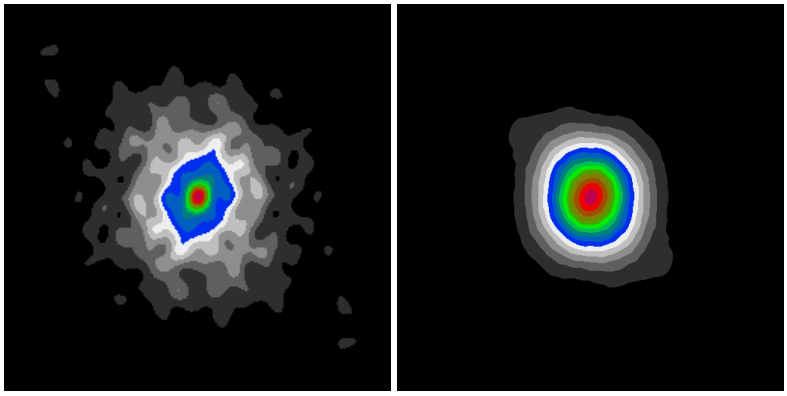
The naive solution is to first update the residuals, and then calculate the correlation with the PSF again. $residuals = residuals - PSF * optimalValue$ $gradients = residuals \star PSF$

But we can simplify this. We can directly update the map of gradients by calculating:

$gradients = gradients - (PSF \star PSF) * optimalValue$

Calculate the PSF correlation with itself. Calculate the PSF correlation once, and use it to update the gradient map directly.

Edges again. We have the problem that, when the PSF is cutoff by the edges of the image, we would need to calculate $(PSF \star PSF)$ again. This is too expensive. Does not change dramatically, unless the pixel we



(a) Point Spread Function. *(b) Gradient update.*

Figure 8: Example problem with two point sources.

optimize is at the full edge.

4.4 Pseudo-code of the basic, optimized algorithm

putting it all together

GPU implementation

4.5 Similarities to the CLEAN algorithm

Comparison to CLEAN. Similar algorithm, but we descend in the actual gradient direction.

4.6 Distributed Deconvolution

How do we distribute the major cycle. We need to distribute every step, Gridding, FFT and Deconvolution.

Gridding, Large number of input data. This needs to be distributed We use the Image domain gridding introduces in sectionand use it as the basis for the distributed gridding.

The FFT is generally not worth distributing, if we can keep all the data in memory. When the gridding is done, in our setup, the grid is small enough to keep in memory. (cite distributed fftw)

Deconvolution is also worth distributing. CLEAN depending on the observation is the second most time consuming step. But gridding tends to be easier to distribute, so in some observations it is the most time consuming step. Split the image into patches and deconvolve each patch. Sadly not possible, we need communication. how we communicate is important.

We use a distributed Gridding and a distributed deconvolution. Which leads us to the following architecture.

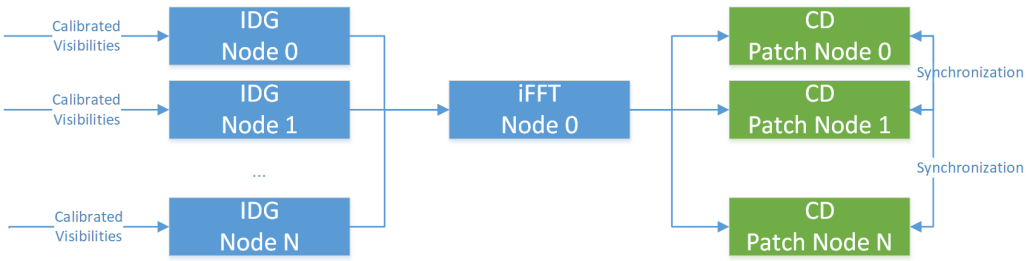


Figure 9: Distributed architecture for half a major cycle

Where each node is one computer, i.e. has its own, possibly multiple cpus and its shared memory. Split the input visibilities onto nodes. Do the gridding locally on each node Communicate the grid inverse FFT on one node. Communicate the patches of the image. Deconvolve each patch and communicate

5 Gradient/PSF approximation for distributed deconvolution

Distributed deconvolution, we would like to deconvovle patches of the final image independently from each other. Concept of seperability. Number of non-zero components.

The size of the PSF tells us how far the patches have to be apart from each other to be independent. Limiting factor for distribution is the size of the PSF. We would like to deconvolve pixels, or patches of the image independently from each other. But the PSF does not let us do that. As soon as the PSF overlap, they are not independent anymore. Generally the PSF spans over the whole image.

But the PSF for MeerKAT is peculiar. In the center, the PSF approximates a Gaussian function, which spans only over a fraction of the image. Indeed, with increasing number of visibility measurements, the PSF becomes more and more like a Gaussian function. Also, the values of the PSF become smaller the further we walk away from the center. Although the PSF spans over the whole image, its values become insiginificant.

MeerKAT has wide field of view observations with an increasingly Gaussian-like PSF. Can we use the significant fraction of the PSF instead? Is this enough? We have two operations, calculating the gradients of the image, and updating the gradients. View of stochastic gradient descent. So if we calculate the gradient of a pixel with $-2\langle residuals, PSF\rangle$, then we should be able to approximate the gradient with only a fraction fo the SF. But the pre-calculation of gradients is actually cheap, the thing. But what about updating the gradients? We

Also, Plus the major cycle framework: It already corrects for the fact that the PSF we use is only an approximation of the true PSF. w-term changes the PSF over the image.

The answer to this is yes, we can. In Section 6.5 we demonstrate the speedup of the deconvolution with only a fraction of the PSF on a real world MeerKAT dataset. In this Section, we show how this can be done.

The problem is arriving at the same optimum. Not trivial.

5.1 Difficulty in approximating the PSF

In our coordinate descent deconvolution, we use the PSF in two steps. Before we run coordinate descent, we pre-calculate the gradient for each pixel with a correlation operation. During coordinate descent deconvolutions, we update the gradient-map directly. Two opposing solutions: We use the full PSF to pre-calculate the gradients, but only update the gradients with a fraction of the PSF. That way we always start from the correct gradients, but get less accurate over coordinate descent deconvolutions.

5.2 Approximate gradient update

So we use the full PSF

only update with a fraction of the PSF

PSF squared update. Scaling. So we become less and less accurate

5.3 Approximate deconvolution

We never use the full PSF

But the problem of gradient magnitude.

Change lambda. We do an approximate deconvolution.

5.4 Major Cycle convergence

[27] and the question on how many iterations per major cycle Putting it all together

We have the Minor Cycle, which is easy to converge.

Coordinate Descent Path optimization [34] Danger that CD takes too many pixel into a Major Cycle. Lower bound per iteration, PSF sidelobe can still be too low, danger when many psf sidelobes overlap

6 Tests on the MeerKAT LMC observation

For algorithmic testing

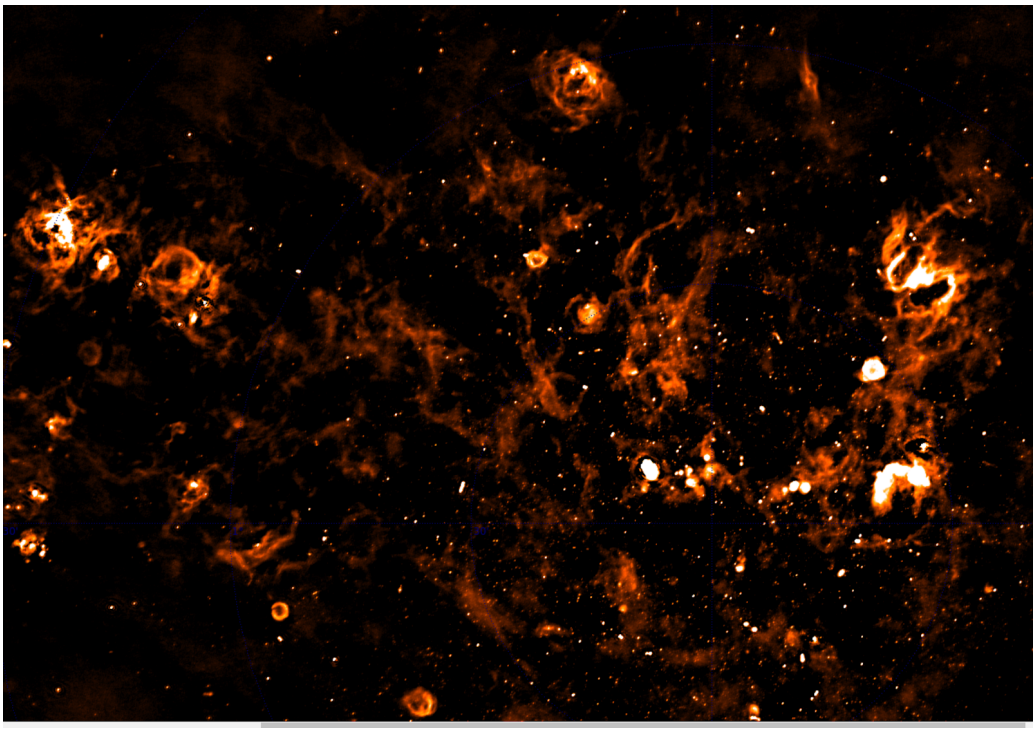


Figure 10: Radio interferometer system

6.1 Comparison with CLEAN reconstruction

6.2 GPU Acceleration

6.3 Distributed coordinate descent

We are in the realm of convolution. Remember that a convolution in image space is a multiplication in fourier space. We can multiply

6.4 Wall clock time

6.5 Validity of gradient approximation

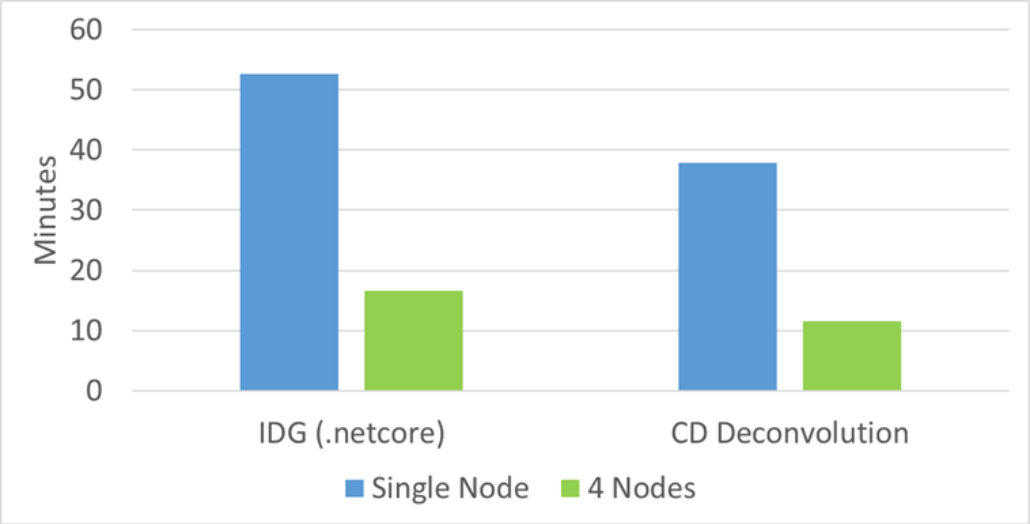


Figure 11: Wall-clock time of the distributed reconstruction

7 Towards parallel coordinate descent

Bells and whistles

- Random
- Block
- Parallel
- Accelerated

PCDM algorithm, and later APPROX.

7.1 Problems

Problem with pure random: each update is fast, but problem with useful updates. −¿ we need blocks where we have a higher chance to update a pixel.

Expansion to BlockCD. Still closed form solution. But numerical problems with closed form. Solution: [39]. We use the gradient step divided by the lipschitz constant.

Towards parallel updates

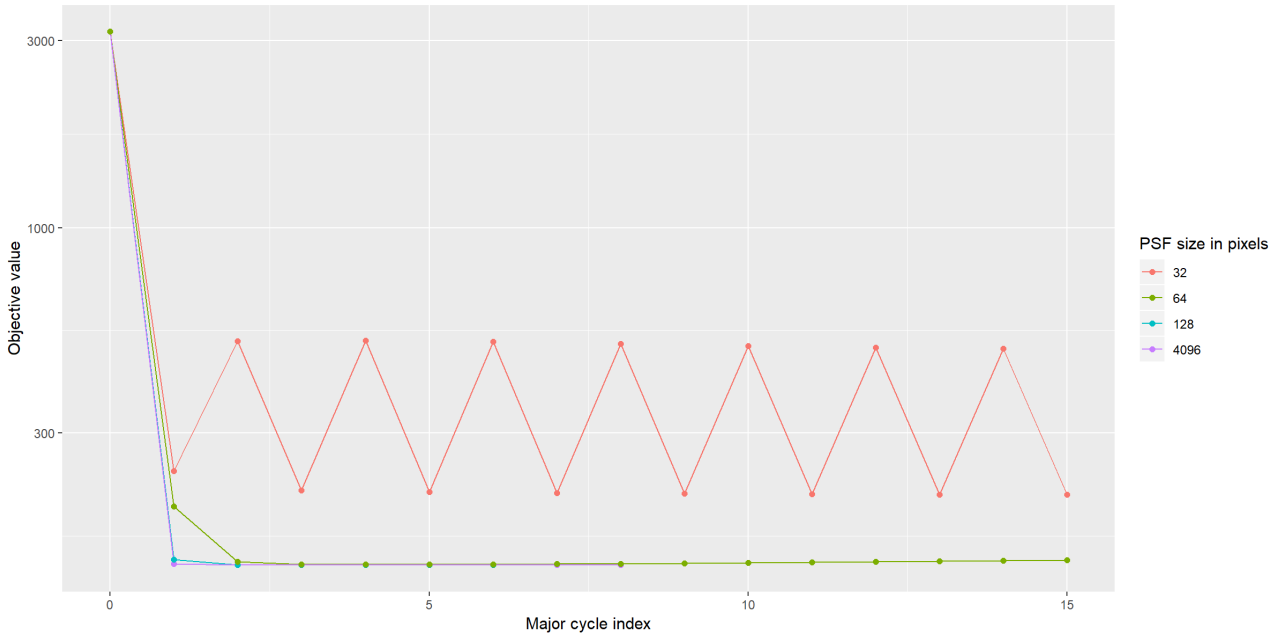


Figure 12: Effect of the L1 and L2 Norm separately.

8 Conclusion

References

[1] Emmanuel J Candès, Justin Romberg, and Terence Tao. Robust uncertainty principles: Exact signal reconstruction from highly incomplete frequency information. IEEE Transactions on information theory, 52(2):489–509, 2006.

[2] David L Donoho. Compressed sensing. IEEE Transactions on information theory, 52(4):1289–1306, 2006.

[3] Arwa Dabbech, Alexandru Onose, Abdullah Abdulaziz, Richard A Perley, Oleg M Smirnov, and Yves Wiaux. Cygnus a super-resolved via convex optimization from vla data. Monthly Notices of the Royal Astronomical Society, 476(3):2853–2866, 2018.

[4] Arwa Dabbech, Chiara Ferrari, David Mary, Eric Slezak, Oleg Smirnov, and Jonathan S Kenyon. More-sane: Model reconstruction by synthesis-analysis estimators-a sparse deconvolution algorithm for radio interferometric imaging. Astronomy & Astrophysics, 576:A7, 2015.

[5] JA Högbom. Aperture synthesis with a non-regular distribution of interferometer baselines. Astronomy and Astrophysics Supplement Series, 15:417, 1974.

[6] Urvashi Rau and Tim J Cornwell. A multi-scale multi-frequency deconvolution algorithm for synthesis imaging in radio interferometry. Astronomy & Astrophysics, 532:A71, 2011.

[7] Charles A Bouman and Ken Sauer. A unified approach to statistical tomography using coordinate descent optimization. IEEE Transactions on image processing, 5(3):480–492, 1996.

[8] Simon Felix, Roman Bolzern, and Marina Battaglia. A compressed sensing-based image reconstruction algorithm for solar flare x-ray observations. The Astrophysical Journal, 849(1):10, 2017.

[9] Madison Gray McGaffin and Jeffrey A Fessler. Edge-preserving image denoising via group coordinate descent on the gpu. IEEE Transactions on Image Processing, 24(4):1273–1281, 2015.

[10] Olivier Fercoq, Zheng Qu, Peter Richtárik, and Martin Takáč. Fast distributed coordinate descent for non-strongly convex losses. In 2014 IEEE International Workshop on Machine Learning for Signal Processing (MLSP), pages 1–6. IEEE, 2014.

[11] Oleg M Smirnov. Revisiting the radio interferometer measurement equation-i. a full-sky jones formalism. Astronomy & Astrophysics, 527:A106, 2011.

[12] Oleg M Smirnov. Revisiting the radio interferometer measurement equation-ii. calibration and direction-dependent effects. Astronomy & Astrophysics, 527:A107, 2011.

[13] Oleg M Smirnov. Revisiting the radio interferometer measurement equation-iii. addressing direction-dependent effects in 21 cm wsrt observations of 3c 147. Astronomy & Astrophysics, 527:A108, 2011.

[14] Oleg M Smirnov. Revisiting the radio interferometer measurement equation-iv. a generalized tensor formalism. Astronomy & Astrophysics, 531:A159, 2011.

[15] Keith Miller. Least squares methods for ill-posed problems with a prescribed bound. SIAM Journal on Mathematical Analysis, 1(1):52–74, 1970.

[16] Yves Wiaux, Laurent Jacques, Gilles Puy, Anna MM Scaife, and Pierre Vandergheynst. Compressed sensing imaging techniques for radio interferometry. Monthly Notices of the Royal Astronomical Society, 395(3):1733–1742, 2009.

[17] André Ferrari, David Mary, Rémi Flamary, and Cédric Richard. Distributed image reconstruction for very large arrays in radio astronomy. In 2014 IEEE 8th Sensor Array and Multichannel Signal Processing Workshop (SAM), pages 389–392. IEEE, 2014.

[18] Julien N Girard, Hugh Garsden, Jean Luc Starck, Stéphane Corbel, Arnaud Woiselle, Cyril Tasse, John P McKean, and Jérôme Bobin. Sparse representations and convex optimization as tools for lofar radio interferometric imaging. Journal of Instrumentation, 10(08):C08013, 2015.

[19] Rafael E Carrillo, Jason D McEwen, and Yves Wiaux. Purify: a new approach to radio-interferometric imaging. Monthly Notices of the Royal Astronomical Society, 439(4):3591–3604, 2014.

[20] Andreas M Tillmann and Marc E Pfetsch. The computational complexity of the restricted isometry property, the nullspace property, and related concepts in compressed sensing. IEEE Transactions on Information Theory, 60(2):1248–1259, 2013.

[21] Abhiram Natarajan and Yi Wu. Computational complexity of certifying restricted isometry property. arXiv preprint arXiv:1406.5791, 2014.

[22] Ishay Haviv and Oded Regev. The restricted isometry property of subsampled fourier matrices. In Geometric Aspects of Functional Analysis, pages 163–179. Springer, 2017.

[23] Emmanuel J Candes and Yaniv Plan. A probabilistic and ripless theory of compressed sensing. IEEE transactions on information theory, 57(11):7235–7254, 2011.

[24] Jean-Luc Starck, David L Donoho, and Emmanuel J Candès. Astronomical image representation by the curvelet transform. Astronomy & Astrophysics, 398(2):785–800, 2003.

[25] Jean-Luc Starck, Fionn Murtagh, and Mario Bertero. Starlet transform in astronomical data processing. Handbook of Mathematical Methods in Imaging, pages 2053–2098, 2015.

[26] Rafael E Carrillo, Jason D McEwen, and Yves Wiaux. Sparsity averaging reweighted analysis (sara): a novel algorithm for radio-interferometric imaging. Monthly Notices of the Royal Astronomical Society, 426(2):1223–1234, 2012.

[27] BG Clark. An efficient implementation of the algorithm'clean'. Astronomy and Astrophysics, 89:377, 1980.

[28] FR Schwab. Relaxing the isoplanatism assumption in self-calibration; applications to low-frequency radio interferometry. The Astronomical Journal, 89:1076–1081, 1984.

[29] AR Offringa and O Smirnov. An optimized algorithm for multiscale wideband deconvolution of radio astronomical images. Monthly Notices of the Royal Astronomical Society, 471(1):301–316, 2017.

[30] Peter Richtárik and Martin Takáč. Distributed coordinate descent method for learning with big data. The Journal of Machine Learning Research, 17(1):2657–2681, 2016.

[31] Peter Richtárik and Martin Takáč. Parallel coordinate descent methods for big data optimization. Mathematical Programming, 156(1-2):433–484, 2016.

[32] Yu Nesterov. Efficiency of coordinate descent methods on huge-scale optimization problems. SIAM Journal on Optimization, 22(2):341–362, 2012.

[33] Yu Nesterov. Gradient methods for minimizing composite functions. Mathematical Programming, 140(1):125–161, 2013.

[34] Jerome Friedman, Trevor Hastie, and Rob Tibshirani. Regularization paths for generalized linear models via coordinate descent. Journal of statistical software, 33(1):1, 2010.

[35] Jason D McEwen and Yves Wiaux. Compressed sensing for wide-field radio interferometric imaging. Monthly Notices of the Royal Astronomical Society, 413(2):1318–1332, 2011.

[36] Shai Shalev-Shwartz and Ambuj Tewari. Stochastic methods for l1-regularized loss minimization. Journal of Machine Learning Research, 12(Jun):1865–1892, 2011.

[37] Olivier Fercoq and Peter Richtárik. Accelerated, parallel, and proximal coordinate descent. SIAM Journal on Optimization, 25(4):1997–2023, 2015.

[38] Jonathan S Kenyon. Pymoresane: Python model reconstruction by synthesis-analysis estimators. Astrophysics Source Code Library, 2019.

[39] Peter Richtárik and Martin Takáč. Iteration complexity of randomized block-coordinate descent methods for minimizing a composite function. Mathematical Programming, 144(1-2):1–38, 2014.

[40] AR Offringa, Benjamin McKinley, Natasha Hurley-Walker, FH Briggs, RB Wayth, DL Kaplan, ME Bell, Lu Feng, AR Neben, JD Hughes, et al. Wsclean: an implementation of a fast, generic wide-field imager for radio astronomy. Monthly Notices of the Royal Astronomical Society, 444(1):606–619, 2014.

[41] Luke Pratley, Melanie Johnston-Hollitt, and Jason D McEwen. A fast and exact w-stacking and w-projection hybrid algorithm for wide-field interferometric imaging. arXiv preprint arXiv:1807.09239, 2018.

[42] Bram Veenboer, Matthias Petschow, and John W Romein. Image-domain gridding on graphics processors. In 2017 IEEE International Parallel and Distributed Processing Symposium (IPDPS), pages 545–554. IEEE, 2017.

List of Figures

List of Tables

9 attachment

10 Larger runtime costs for Compressed Sensing Reconstructions

The MeerKAT instrument produces a new magnitude of data volume. An image with several million pixels gets reconstructed from billions of Visibility measurements. Although MeerKAT measures a large set of Visibilities, the measurements are still incomplete. We do not have all the information available to reconstruct an image. Essentially, this introduces ”fake” structures in the image, which a reconstruction algorithm has to remove. Additionally, the measurements are noisy.

We require an image reconstruction algorithm which removes the ”fake” structures from the image, and removes the noise from the measurements. The large data volume of MeerKAT requires the algorithm to be both scalable and distributable. Over the years, several reconstruction algorithms were developed, which can be separated into two classes: Algorithms based on CLEAN, which are cheaper to compute and algorithms based on Compressed Sensing, which create higher quality reconstructions.

CLEAN based algorithms represent the reconstruction problem as a deconvolution. First, they calculate the ”dirty” image, which is corrupted by noise and fake image structures. The incomplete measurements essentially convolve the image with a Point Spread Function (PSF). CLEAN estimates the PSF and searches for a deconvolved version of the dirty image. In each CLEAN iteration, it searches for the highest pixel in the dirty image, subtracts a fraction PSF at the location. It adds the fraction to the same pixel location of a the ”cleaned” image. After several iterations, the cleaned image contains the deconvolved version of the dirty image. CLEAN accounts for noise by stopping early. It stops when the highest pixel value is smaller than a certain threshold. This results in a light-weight and robust reconstruction algorithm. CLEAN is comparatively cheap to compute, but does not produce the best reconstructions and is difficult to distribute on a large scale.

Compressed Sensing based algorithms represent the reconstruction as an optimization problem. They search for the optimal image which is as close to the Visibility measurements as possible, but also has the smallest regularization penalty. The regularization encodes our prior knowledge about the image. Image structures which were likely measured by the instrument result in a low regularization penalty. Image structures which were likely introduced by noise or the measurement instrument itself result in high penalty. Compressed Sensing based algorithms explicitly handle noise and create higher quality reconstructions than CLEAN. State-of-the-art Compressed Sensing algorithms show potential for distributed computing. However, they currently do not scale on MeerKATs data volume. They require too many computing resources compared to CLEAN based algorithms.

This project searches for a way to reduce the runtime costs of Compressed Sensing based algorithms. One reason for the higher costs is due to the non-uniform FFT Cycle. State-of-the-art CLEAN and Compressed Sensing based algorithms both use the non-uniform FFT approximation in a cycle during reconstruction. The interferometer measures the Visibilities in a continuous space in a non-uniform pattern. The image is divided in a regularly spaced, discrete pixels. The non-uniform FFT creates an approximate, uniformly sampled image from the non-uniform measurements. Both, CLEAN and Compressed Sensing based algorithms use the non-uniform FFT to cycle between non-uniform Visibilities and uniform image. However, a Compressed Sensing algorithm requires more non-uniform FFT cycles for reconstruction.

CLEAN and Compressed Sensing based algorithms use the non-uniform FFT in a similar manner. However, there are slight differences in the architecture. This project hypothesises that The previous project searched for an alternative to the non-uniform FFT cycle. Although there are alternatives, there is currently no replacement which leads to lower runtime costs for Compressed Sensing. Current research is focused on reducing the number of non-uniform FFT cycles for Compressed Sensing algorithms.

CLEAN based algorithms use the Major Cycle Architecture for reconstruction. Compressed Sensing based algorithms use a similar architecture, but with slight modifications. Our hypothesis is that we may reduce the number of non-uniform FFT cycles for Compressed Sensing by using CLEAN’s Major Cycle Architecture.

10.1 CLEAN: The Major Cycle Architecture

Figure 14 depicts the Major Cycle Architecture used by CLEAN algorithms. First, the Visibilities get transformed into an image with the non-uniform FFT. The resulting dirty image contains the corruptions of the measurement instrument and noise. A deconvolution algorithm, typically CLEAN, removes the corruption of the instrument with a deconvolution. When the deconvolution stops, it should have removed most of the observed structures from the dirty image. The rest, mostly noisy part of the dirty image gets transformed back into residual Visibilities and the cycle starts over.

In the Major Cycle Architecture, we need several deconvolution attempts before it has distinguished the noise from the measurements. Both the non-uniform FFT and the deconvolution are approximations. By using the non-uniform FFT in a cycle, it can reconstruct an image at a higher quality. For MeerKAT reconstruction with CLEAN, we need approximately 4-6 non-uniform FFT cycles for a reconstruction.

10.2 Compressed Sensing Architecture

Figure 15 depicts the architecture used by Compressed Sensing reconstructions. The Visibilities get transformed into an image with the non-uniform FFT approximation. The algorithm then modifies the image so it reduces the regularization penalty. The modified image gets transformed back to Visibilities and the algorithm then minimizes the difference between measured and reconstructed Visibilities. This is repeated until the algorithm converges to an optimum.

In this architecture, state-of-the-art Compressed Sensing algorithms need approximately 10 or more non-uniform FFT cycles to converge. It is one source for the higher runtime costs. For MeerKAT reconstructions the non-uniform FFT tends to dominate the runtime costs. A CLEAN reconstruction with the Major Cycle Architecture already spends a large part of its time in the non-uniform FFT. Compressed Sensing algorithms need even more non-uniform FFT cycle on top of the ”Image Regularization” step being generally more expensive than CLEAN deconvolution. There is one upside in this architecture: State-of-the-art algorithms managed to distribute the ”Image Regularization” operation.

10.3 Hypothesis for reducing costs of Compressed Sensing Algorithms

Compressed Sensing Algorithms are not bound to the Architecture presented in section 10.2. For example, we can design a Compressed Sensing based deconvolution algorithm and use the Major Cycle Architecture instead.

Our hypothesis is: We can create a Compressed Sensing based deconvolution algorithm which is both distributable and creates higher quality reconstructions than CLEAN. Because it also uses the Major Cycle architecture, we reckon that the Compressed Sensing deconvolution requires a comparable number of non-uniform FFT cycles to CLEAN. This would result in a Compressed Sensing based reconstruction algorithm with similar runtime costs to CLEAN, but higher reconstruction quality and higher potential for distributed computing.

10.4 State of the art: WSCLEAN Software Package

10.4.1 W-Stacking Major Cycle

10.4.2 Deconvolution Algorithms

CLEAN MORESANE

10.5 Distributing the Image Reconstruction

10.5.1 Distributing the Non-uniform FFT

10.5.2 Distributing the Deconvolution

11 Handling the Data Volume

The new data volume is a challenge to process for both algorithms and computing infrastructure. Push for parallel and distributed algorithms. For Radio Interferometer imaging, we require specialized algorithms. The two distinct operations, non-uniform FFT and Deconvolution, were difficult algorithms for parallel or distributed computing.

The non-uniform FFT was historically what dominated the runtime []. Performing an efficient non-uniform FFT for Radio Interferometers is an active field of research[40, 41], continually reducing the runtime costs of the operation. Recently, Veeneboer et al[42] developed a non-uniform FFT which can be fully executed on the GPU. It speeds up the most expensive operation.

In Radio Astronomy, CLEAN is the go-to deconvolution algorithm. It is light-weight and compared to the non-uniform FFT, a cheap algorithm. It is also highly iterative, which makes it difficult for effective parallel or distributed implementations. However, compressed sensing based deconvolution algorithms can be developed with distribution in mind.

11.1 Fully distributed imaging algorithm

Current imaging algorithms push towards parallel computing with GPU acceleration. But with Veeneboer et al's non-uniform FFT and a compressed sensing based deconvolution, we can go a step further and create a distributed imaging algorithm.

12 Image Reconstruction for Radio Interferometers

In Astronomy, instruments with higher angular resolution allows us to measure ever smaller structures in the sky. For Radio frequencies, the angular resolution is bound to the antenna dish diameter, which puts practical and financial limitations on the highest possible angular resolution. Radio Interferometers get around this limitation by using several smaller antennas instead. Together, they act as a single large antenna with higher angular resolution at lower financial costs compared to single dish instruments.

Each antenna pair of an Interferometer measures a single Fourier component of the observed image. We can retrieve the image by calculating the Fourier Transform of the measurements. However, since the Interferometer only measures an incomplete set of Fourier components, the resulting image is ”dirty”, convolved with a Point Spread Function (PSF). Calculating the Fourier Transform is not enough. To reconstruct the from an Interferometer image, an algorithm has to find the observed image with only the dirty image and the PSF as input. It has to perform a deconvolution. The difficulty lies in the fact that there are potentially many valid deconvolutions for a single measurement, and the algorithm has to decide for the most likely one. How similar the truly observed image and the reconstructed images are depends largely on the deconvolution algorithm.

State-of-the-art image reconstructions use the Major Cycle architecture (shown in Figure 16), which contains three operations: Gridding, FFT and Deconvolution.

The first operation in the Major Cycle, Gridding, takes the non-uniformly sampled Fourier measurements from the Interferometer and interpolates them on a uniformly spaced grid. The uniform grid lets us use FFT to calculate the inverse Fourier Transform and we arrive at the dirty image. A deconvolution algorithm takes the dirty image plus the PSF as input, producing the deconvolved ”model image”, and the residual image as output. At this point, the reverse operations get applied to the residual image. First the FFT and then De-gridding, arriving at the non-uniform Residuals. The next Major Cycle begins with the non-uniform Residuals as input. The cycles are necessary, because the Gridding and Deconvolution operations are only approximations. Over several cycles, we reduce the errors introduced by the approximate Gridding and Deconvolution. The final, reconstructed image is the addition of all the model images of each Major Cycle.

12.1 Distributed Image Reconstruction

New Interferometer produce an ever increasing number of measurements, creating ever larger reconstruction problems. A single image can contain several terabytes of Fourier measurements. Handling reconstruction problems of this size forces us to use distributed computing. However, state-of-the-art Gridding and Deconvolution algorithms only allow for limited distribution. How to scale the Gridding and Deconvolution algorithms to large problem sizes is still an open question.

Recent developments make a distributed Gridder and a distributed Deconvolution algorithm possible. Veeneboer et al[42] found an input partitioning scheme, which allowed them to perform the Gridding on the GPU. The same partitioning scheme can potentially be used to distribute the Gridding onto multiple machines. For Deconvolution, there exist parallel implementations for certain algorithms like MORESANE[4]. These can be used as a basis for a fully distributed image reconstruction.

In this project, we want to make the first steps towards an image reconstruction algorithm, which is distributed from end-to-end, from Gridding up to and including deconvolution. We create our own distributed Gridding and Deconvolution algorithms, and analyse the bottlenecks that arise.

12.2 First steps towards a distributed Algorithm

In this project, we make the first steps towards a distributed Major Cycle architecture (shown in figure 16) implemented C#. We port Veeneboer et al’s Gridder, which is written in C++, to C# and modify it for distributed

computing. We implement a simple deconvolution algorithm based on the previous project and create a first, non-optimal distributed version of it.

In the next step, we create a more sophisticated deconvolution algorithm based on the shortcomings of the first implementation. We use simulated and real-world observations of the MeerKAT Radio Interferometer and measure its speed up. We identify the bottlenecks of the current implementation and explore further steps.

From the first lessons, we continually modify the distributed algorithm and focus on decreasing the need for communication between the nodes, and increase the overall speed up compared to single-machine implementations. Possible Further steps:

- Distributed FFT
- Replacing the Major Cycle Architecture
- GPU-accelerated Deconvolution algorithm.

A state-of-the-art reconstruction algorithm has to correct large number of measurement effects arising from the Radio Interferometer. Accounting for all effects is out of the scope for this project. We make simplifying assumptions, resulting in a proof-of-concept algorithm.

13 Ehrlichkeitserklärung

Hiermit erkläre ich, dass ich die vorliegende schriftliche Arbeit selbstständig und nur unter Zuhilfenahme der in den Verzeichnissen oder in den Anmerkungen genannten Quellen angefertigt habe. Ich versichere zudem, diese Arbeit nicht bereits anderweitig als Leistungsnachweis verwendet zu haben. Eine Überprüfung der Arbeit auf Plagiate unter Einsatz entsprechender Software darf vorgenommen werden.
Windisch, October 8, 2019

Jonas Schwammberger

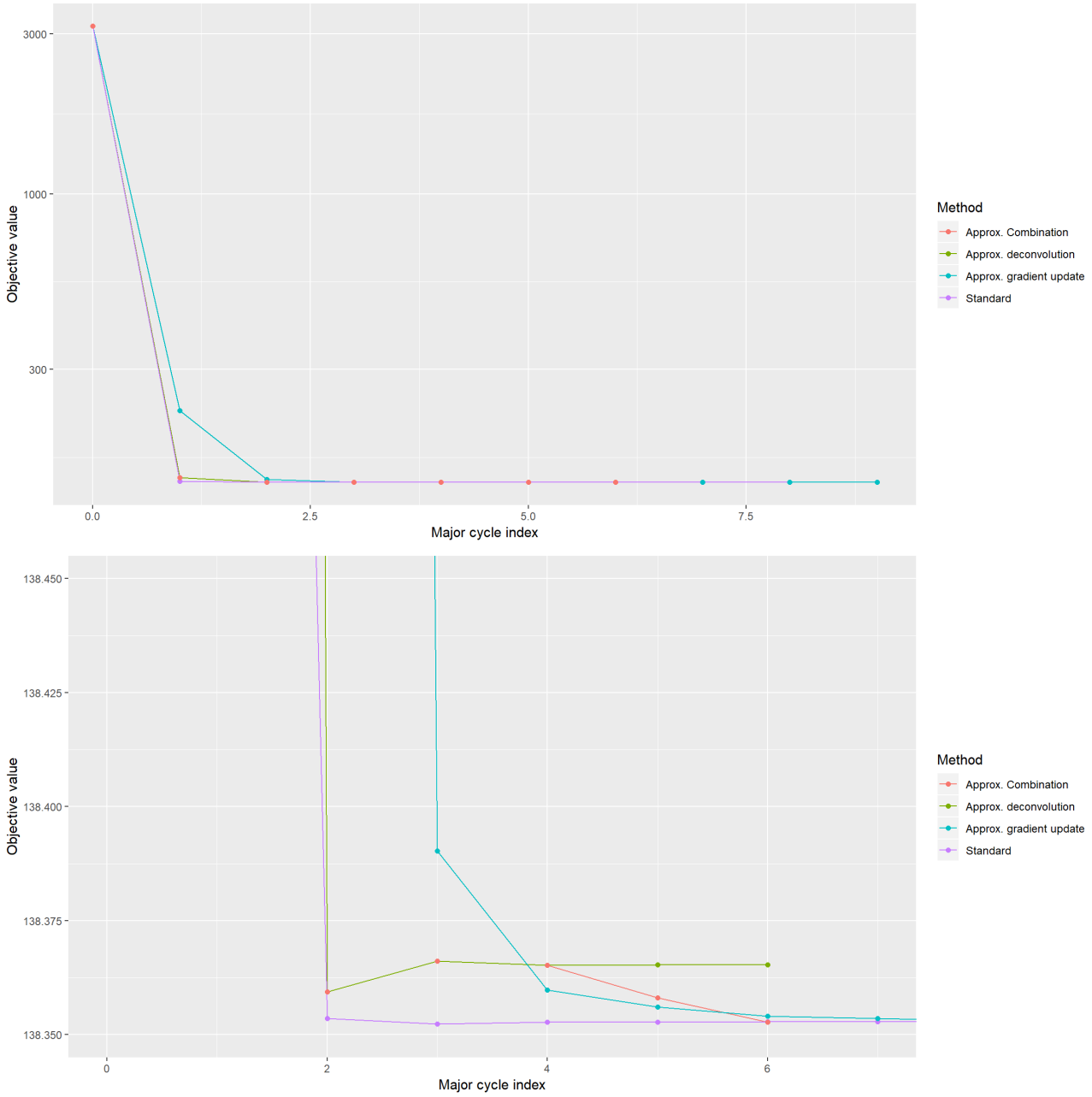


Figure 13: Effect of the L1 and L2 Norm separately.

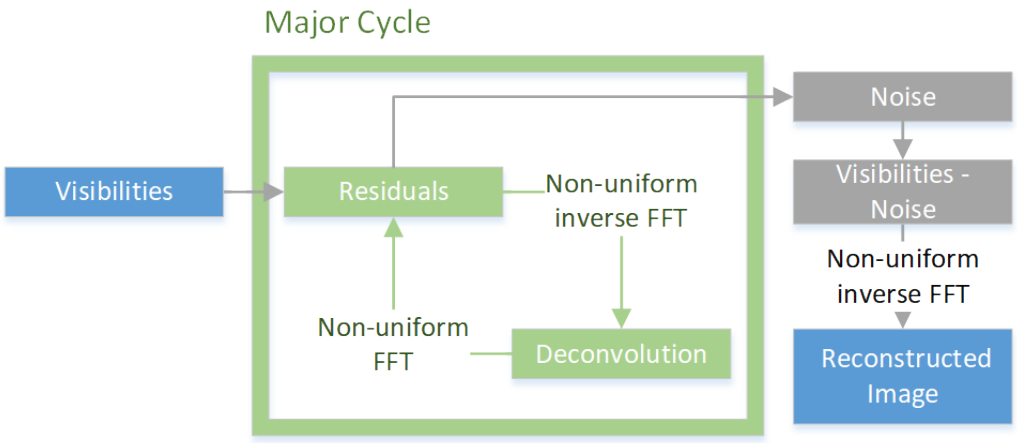


Figure 14: The Major Cycle Architecture

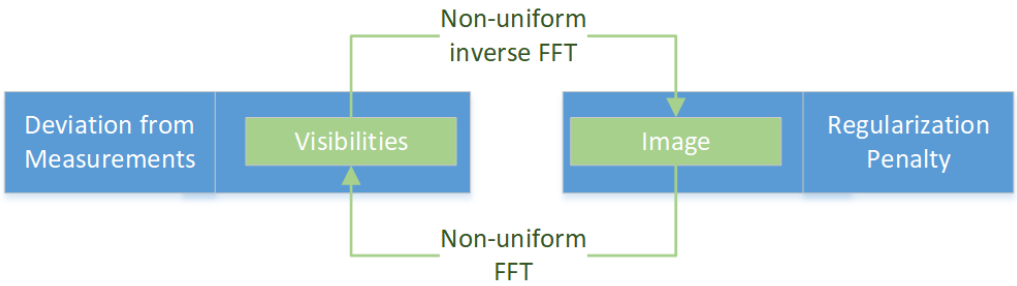


Figure 15: State-of-the-art Compressed Sensing Reconstruction Architecture

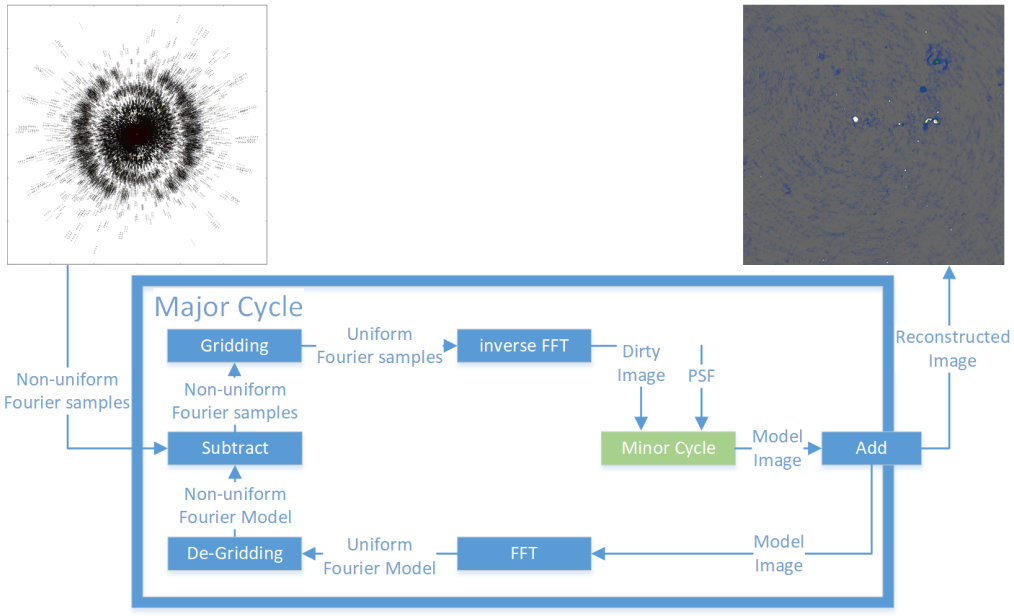


Figure 16: The Major Cycle Architecture of image reconstruction algorithms