P8 Exploring new Architectures for Compressed Sensing Reconstructions in the MeerKAT era

Jonas Schwammberger

January 27, 2019

Abstract

MeerKAT new Radio Interferometer, poses a large scale image reconstruction problem.

Compressed sensing reconstructions for Radio astronomy exist, but are slower than CLEAN.

Compressed Sensing and CLEAN use the same architecture, both use Major Cycle. In this work, we explore if Compressed Sensing reconstructions can be made faster with in a different architecture

Several possibiliites are discussed and a new proof-of-concept algorithm is developed, which does not require a Major Cycle. We estimate the minimum runtime complexity of the algorithm and compare it to CLEAN.

However, the algorithm is not suitable in the context of meerkat reconstructions. It requires too much memory and

we come to the conclusion that in the context of MeerKAT reconstructions, the Major Cycle architecture is easier to compute.

No obvious contender to replace the Major Cycle.

Contents

1 Compressed Sensing Image Reconstruction for MeerKAT

An instrument in the real world measures noisy data. Measurements are corrupted by noise, interference sources or the measurement instrument itself. Image reconstruction problems appear when one tries to remove the corruption from the measurements and tries to find the observed image of the instrument. This leads to an ill-posed inverse problem: A small change in the measurements may create very different image reconstructions, and many possible images match the measurements. An image reconstruction algorithm therefore has to find the observed image from a potentially large set of possible images.

In the past, image reconstructions applied simple heuristics and approximated a likely image. How close the approximation was to the observed image was in general not known. The theory of compressed sensing[1][2] introduced a new theoretical framework under which image reconstructions can be analysed. This has lead rise to new compressed sensing reconstruction algorithms, which under the right conditions are guaranteed to reconstruct the observed image. Furthermore they have shown super-resolution performance in real-world environments, creating reconstructions above the accuracy limit of the instrument.

This work applies compressed sensing image reconstruction to the field of radio astronomy. The new MeerKAT radio interferometer poses a reconstruction problem on a new scale of data volume. The raw measurements easily take up several terabytes of disk space. The focus of this work is creating a scalable compressed sensing image reconstruction algorithm.

The current state of the art reconstruction algorithm for MeerKAT is based on CLEAN[3][4]. It is a reconstruction using a simple heuristic and was developed before the theory of compressed sensing was known. In recent years, compressed sensing reconstruction algorithms were developed for radio interferometers[5][6][7]. They beat CLEAN in terms of reconstruction quality, producing super-resolved reconstructions. However, CLEAN has the upper hand in runtime complexity. Therefore, an efficient implementation of CLEAN is still the go to reconstruction algorithm for MeerKAT data.

The efficient implementation use CLEAN in the Major Cycle Architecture, which was developed with CLEAN in mind. Current compressed sensing algorithms use a similar architecture. So far, little research has gone into different architectures for compressed sensing reconstructions in radio astronomy. This work explores different architectures for compressed sensing reconstructions with the hope of reducing the computational complexity.

A new proof-of-concept reconstruction algorithm was developed with a simplified architecture. It was tested on simulated MeerKAT data and the lower-bound asymptotic complexity was evaluated. The new algorithm scales independently of the image size, but scales worse with the number of input measurements compared to CLEAN. For the MeerKAT image reconstruction, the number of input measurements tends to be the largest of all numbers in the problem. Even though the algorithm could be improved further, it is unlikely to beat CLEAN reconstructions in terms of runtime complexity on MeerKAT data.

1.1 The basic Measurement Equation of a radio interferometry

Real world radio interferometers have complicated measurement equations. They become even more complicated for large interferometers like MeerKAT. These problems get addressed in section 2. This section looks at the basic measurement equation of a radio interferometer (1.1) and discusses the two fundamental challenges for radio interferometry image reconstruction.

$$V(u,v) = \int\int I(x,y) e^{2\pi i(ux+vy)}\, dx\, dy \tag{1.1}$$

An interferometer measures Fourier Components V (called Visibilities in Radio Astronomy) from the sky image I at position x and y. The term $e^{2\pi i(ux+vy)}$ represents the two dimensional Fourier Transform. The task is to reconstruct the observed image I from the measured Visibilities V. In theory this task is trivial: Since the inverse Fourier Transform exists, we can reconstruct the image I by calculating the inverse Fourier Transform of V. However, two properties of the Visibilities make this task challenging in practice:

1. Non-uniform sampling pattern in Visibility space
2. Incomplete Visibility coverage.

Property 1: We want to reconstruct an image with uniformly spaced pixels. The instrument defines the sampling pattern in Visibility space and does not correspond to the exact pixels of the reconstructed image. This property keeps us from using the Fast Fourier Transform. The naive inverse Fourier Transform can still be calculated, but it has a quadratic runtime and does not scale to the data volume of interferometers. Current reconstruction algorithms use the non-uniform Fast Fourier Transform. The non-uniform FFT approximates the non-uniform Fourier Transform.

Property 2: Interferometers sample only a limited set of Visibilities. It does not have all information for reconstruction. When the inverse Fourier Transform is applied on the Visibilities, the resulting image is corrupted by the incomplete Visibilities. It contains structures which where introduced by the interferometer and were not observed. With only knowing the incomplete set of Visibilities a reconstruction algorithm has to decide which image structures were truly measured, and which are due to the instrument. This forms an ill-posed inverse problem. There are many images that fit the measurements, and a small change in the Visibilities can lead to a very different reconstruction.

CLEAN represents the instrumental effect with a Point Spread Function (PSF). After the non-uniform FFT produced the 'dirty' image, CLEAN tries to reconstruct observed image with a deconvolving the 'dirty' image with the PSF. Note however that both CLEAN nor the non-uniform FFT are approximations. In real world reconstructions, these two approximations are used in the major cycle architecture to increase the reconstruction accuracy.

1.2 The Major Cycle Architecture

Major cycle was created with CLEAN in mind. Compressed sensing reconstructions use essentially the same architecture with minor modifications.

A CLEAN image reconstruction for radio interferometers consists of two different steps: A non-uniform FFT, which approximates the inverse Fourier Transform efficiently and an deconvolution algorithm, which approximates the instrumental effects on the image (typically based on CLEAN). These two approximations are an error source of the reconstruction. The major cycle architecture 1 therefore tries to iteratively minimize the errors of the non-uniform FFT and the deconvolution over several iterations.

The first major cycle uses the non-uniform inverse FFT to approximate the 'dirty' image from the Visibilities. The deconvolution algorithm decides parts which parts belong to the observed image and which are due to instrumental and other noise effects. It returns the noise part of the image, which gets transformed back into residual Visibilities. The next major cycle iteration continues with the

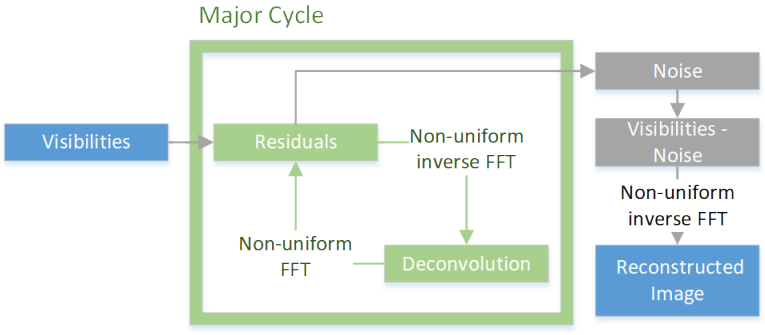


Figure 1: The Major Cycle Framework

residual Visibilities. The residual Visibilities get minimized until they contain only the instrumental effects and noise.

After several major cycles the residual on a regularly spaced image which has a small error from non-uniform samples, and a small error from incomplete measurements.

Compressed sensing reconstructions essentially use the same architecture, although with minor alterations. In general, they keep a similar scheme of forward/backward non-uniform FFT, but do not use a deconvolution in image pace. Instead, they analyse constraints defined on the image (For example all pixels should be non-negative), and try to find the Visibilities that minimize the constraint violations.

For CLEAN reconstructions, the forward/backward non-uniform FFT are the most expensive operations, while the deconvolution is negligible. The compressed sensing algorithms tend to require more major cycle iterations to converge and the image constraint analysis tends to be more expensive than CLEAN deconvolutions. Current research into compressed sensing reconstructions is focussed on reducing the number of major cycles[6].

2 Challenges for imaging MeerKAT data

New interferometers like MeerKAT put additional challenges on the image reconstruction problem. First, the new instrument produce a new magnitude of data, forcing reconstruction algorithms to be highly scalable and distributable. Second, the more sensitive instruments with large field of view amplify effects which were negligible in older instruments, like non-coplanar baselines or the ionosphere.

Compressed sensing has to be able to work with these issues somehow. Solutions all in the context of the major cycle, a new architecture may need to handle effects differently. Hopefully in a manner that is at least as efficient.

In this work, the effects of non-coplanar baselines gets handled in more detail. The effect has a neat mathematical notation, it adds a third fourier component to the measurement equation, but breaks the two dimensional fourier relationship introduced in section 1.1.

Calibration used to be a task before image reconstruction. Calibration gets not explicitly handled here, but it is important to keep in mind. For older interferometers, the data was calibrated before image reconstruction. With the advent of self-calibration, the image reconstruction is also used to improve calibration. MeerKAT requires more calibration parameters, so an image reconstruction algorithm has to be able to calibrate, or it is not interesting.

Further issues that do not get handled here

- (Beam Pattern, A Projection)
- Full polarization
- Wide band imaging

There are several challenger for imaging meerkat data. One problem is the new amount of data.

terabytes of measurements. Large image size 32k squared are the obvious problems to solve. Distributing the problem is not part of this work.

In this work, it is focused on Wide field of view issue.

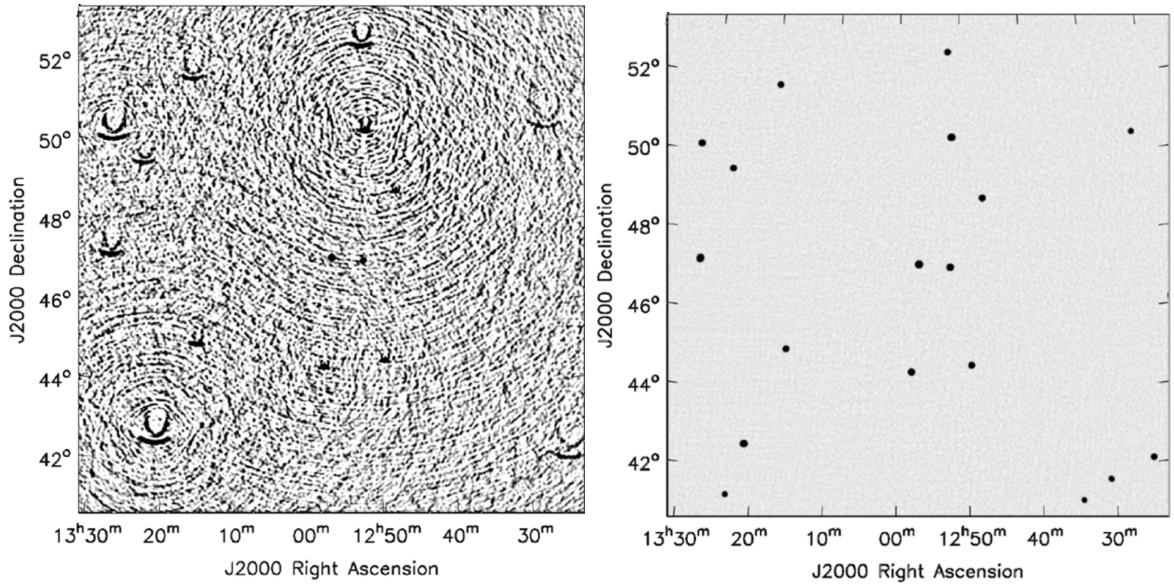
2.1 Wide Field of View Imaging and the third Fourier Component

In wide field of view imaging, the simplifications we could make from the basic measurement equation (1.1) do not hold. The Visibility space of an interferometer actually has a third w-component. This leads us to the wide field of view measurement equation (2.1)

$$V(u,v,w) = \int\int \frac{I(x,y)}{\sqrt{1-x^2-y^2}} e^{2\pi i[ux+vy+w(\sqrt{1-x^2-y^2}-1)]}\, dx\, dy \tag{2.1}$$

For a small field of view, the term $\sqrt{1-x^2-y^2} \approx 1$, which simplifies to our original (1.1), we can ignore the w-term and use the two dimensional Fourier transform. For wide field of view however, the w-component cannot be ignored. The figure 2b shows the effect of ignoring w. The image gets continually more distorted away from the center. Emissions at the edges of the image get ”torn” apart.

We find the reason for why it distorts the edges, if we look at the measurement equation (2.1) in detail. We see that the image $I(x,y)$ still two dimensional, even with a three dimensions in Visibility space. The observed image of the interferometer is not on a flat plane, but on a curved surface(hence the w-component)[9]. More precisely, the instrument looks at the inside wall of the celestial sphere. The two dimensional Fourier transform



(a) 2D Fourier Transform. *(b) Fourier Transform with w-correction.*

Figure 2: Celestial sphere distortion on simulated data. Source: [8]

approximates the sphere with a flat plane, where the tangent point is typically the image center. The further away we move from the tangent point, the more distortion gets added by the curvature, represented by the term $\sqrt{1-x^2-y^2}$. The curve adds a phase shift the further away we move from the tangent point. If the reconstruction algorithm ignores the w-component for a wide field of view, the phase-shift gets severe enough to decorrelate the Visibilities, "tearing" apart the image structures of (2).

In the context of MeerKAT, the reconstruction algorithm has to handle the w-term in an efficient manner. The third Fourier breaks the two dimensional relationship between image and Visibilities, which keeps us from using the non-uniform FFT. In the Major Cycle architecture, w-stacking is the state-of-the-art way of correcting for the w-component efficiently. Since this project explores different architecture, we may need to find another way of correcting for the w-term.

2.1.1 State of the art: w-stacking algorithm and WSCLEAN

w-stacking[10] is the current state-of-the-art approach for w-corrections. It is implemented in the WSCLEAN software package and is the current go-to algorithm for MeerKAT reconstructions. With w-stacking, we can again use the two dimensional non-uniform FFT for efficient transformations between image and Visibilities.

w-stacking, shown in figure 3 replaces the non-uniform FFT operation from figure 1. It groups the Visibilities with similar w-term into the same layer. Multiple layers are then stacked and transformed independently with the non-uniform FFT. For each stack, the w-correction is performed for the center w-component. The last step is summing over all stacked images, giving us the final dirty image. At this point a deconvolution algorithm like CLEAN can deconvolve the dirty image.

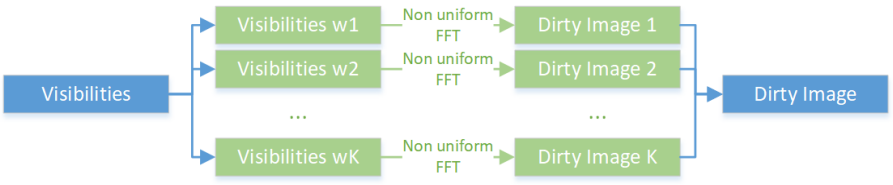


Figure 3: The Major Cycle Framework

After the deconvolution, the whole process is reversed. The image gets copied back into the stacks, the w-correction gets reversed and the non-uniform FFT calculates the Visibilities per stack.

If we choose uses as many stacks as Visibilities, we end up with an exact w-corrections. In pracice, many Visibilities have a similar w-term. w-stacking approximates the correction and allows the Major Cycle to distribute the forward- and backwards Fourier transform to a certain extend.

2.2 Self-Calibration

Radio Interferometer require calibration of their complex valued Visibility measurements. The amplitude and phase of a Visibility is subject to

Before the advent of self-calibration, the calibration terms were Complex gain term. Corrects amplitude and phase.

Traditional Calibration

A calibration source close by, with a known brightness. Phase and amplitude calibration was done before image reconstruction. For older interferometers, there was a very limited number of calibration terms to solve.

but again effects of wide field of view also increase the number of necessary calibration terms. The advent of self calibration, in which the image reconstruction was used to solve for both, the observed image and the calibration of the instrument.

Self calibration with the major cycle algorithm and a CLEAN deconvolution.

Initial phase calibration Shallow clean phase calibration deep clean phase and amplitude calibration deep clean reconstruction

3 Eliminating the Major Cycle

Although Compressed Sensing algorithms produce higher quality reconstructions, the CLEAN algorithm so far has lower runtime costs. Both use a version of the Major Cycle architecture for efficient transformation of Visibilities to image space, and to correct the effects of wide field of view imaging. This project explores different architectures for Compressed Sensing, with the goal to decrease the runtime costs in the context of MeerKAT.

Many Compressed Sensing reconstructions use the Major Cycle architecture[5, 6, 9, 10] and do not discuss alternatives. Runtime improvements are explored in the context of the Major Cycle. Pratley et al[11] optimized the non-uniform FFT in the context of Compressed Sensing reconstructions. Dabbech et al.[13] investigated w-term approximations for reducing runtime costs of Compressed Sensing reconstructions.

An alternative to the Major Cycle was investigated by Hardy et al.[12]. They replaced the non-uniform FFT with the direct Fourier transform and distributed the problem.

We

We want explore further and discuss. why they don't work. In this section, we discuss three possible alternatives to the Major Cycle: Separating the Major Cycle in two Optimization problems, using Spherical Harmonics,

3.1 Separating the Major Cycle into two Optimization Problems

In each iteration, the Major Cycle reduces the error from both, the non-uniform FFT approximation, and from incomplete measurements. Here, we discuss the potentials and problems of separating the problem into two optimization objectives (3.1) (3.2) and solve each after the other. This architecture only needs the dirty image for further processing, which is several times smaller than the original Visibility measurements.

$$\underset{I_{dirty}}{minimize} \left\| V - \tilde{F} I_{dirty} \right\|_2^2 \tag{3.1}$$

$$\underset{x}{minimize} \left\| I_{dirty} - x \star PSF \right\|_2^2 + \lambda \left\| P(x) \right\|_1 \tag{3.2}$$

The first objective (3.1) uses the non-uniform FFT approximation $\tilde{F}$ and searches for the optimal dirty image matching the Visibilities. The optimization algorithm for the first objective looks similar to the Major Cycle architecture. It still uses the non-uniform FFT approximation, and iteratively transforms the Visibilities to image space and back. In this architecture, we can use specialized optimization algorithms potentially uses fewer non-uniform FFT approximations to converge to the dirty image. At this point, we can drop the original Visibility measurements and only use the dirty image for further processing. The error introduced by the incomplete set of Visibilities can be represented as a convolution in image space with a Point Spread Function. The second objective (3.2) therefore can minimized to reconstruct the observed image x, which just needs the dirty image and the PSF.

The downside of this architecture lies in the PSF of the second objective. The PSF varies over the image. For the most accurate result, we would need a PSF for each pixel. A single PSF is calculated from our Visibilities, where all amplitudes are 1 and phase-shifted to the desired position. For every PSF calculation we need another non-uniform FFT approximation, leading to a quadratic runtime.

CLEAN gets away with a constant PSF[1], because of the Major Cycle architecture. In each cycle, it can reduce the error it introduced in the previous cycle, leading to a more accurate reconstruction. In the new architecture, we need several PSFs for a comparable accuracy to the Major Cycle, and this number dictates whether we can reduce the runtime costs. The question, how many PSFs are necessary, seems currently unknown in the Radio Astronomy community, as it becomes irrelevant for the Major Cycle. As of the time of writing, we do not know of any published research into this area. We do not have good estimates on the viability of this architecture.

The main advantage for this architecture is that it reduces the problem to a uniformly-sampled grid, and ignore the original Visibilities in later steps. However, in the context of self-calibration, we cannot ignore the non-uniformly sampled Visibilities. We need a transformation from the image back to the original Visibilities and improve the calibration parameters. Self-calibration negates the main advantage of the architecture. Unless we find a way to map the Visibilities together with their calibration parameters on a uniformly-sampled grid, this architecture does not seem practical for MeerKAT.

3.2 Spherical Harmonics

Spherical Harmonics are a different way to represent the measurements of an interferometer. Carozzi[14] derived a measurement equation based on the fact that the Visibilities fulfils the Helmholtz equation. In practical terms, Carozzi showed we can replace the non-uniform FFT with the Spherical Harmonics Transform in the Major Cycle architecture.

Deconvolution algorithms like CLEAN are still possible, but the true potential is reconstructing on the sphere directly. MCewen et al.[15] showed how to do wavelets on the sphere and used it to speed up the simulation of Visibilities.

This is the only work in that area. The rest is focussed on improving the reconstruction quality instead of speed.[9]

It is unknown, maybe there is a way to improve the runtime, but so far nobody has published anything.

If there is, it may be difficult to translate calibration, ionosphere and the stuff to the spherical harmonics measurement equation.

We have had decades of Fourier-related experience and what effects they do. RIME

Proof of concept implementations. Not a reconstruction algorithm with Spherical Harmonics transform which handles a realitic instrument.

3.3 Direct Fourier Transform improvement

Simplify realistic

Hardy et al[12] handled the whole inverse Fourier Transform as an explicit matrix F^{-1} of size $M * N$. They dropped the non-uniform FFT and w-stacking approximations. This is trivially to distribute later. The only problem is of Course that for MeerKAT data sizes $M * N$ is too large ().

However there is potential for improvement: Only a limited number of pixels in a reconstruction are not zero. We spent a lot of processing power on pixels which do not matter in the reconstruction. If we could predict which pixels are not zero, we would only need to calculate a subset of F^{-1} columns to reconstruct the

[1] CLEAN implementations in Radio Astronomy typically use a constant PSF. This is not necessary, one can also implement CLEAN with a varying PSF.

image. The starlet transform[19] has a way to predict its non-zero components. We represent the image as a combination of starlets, and estimate which starlets are likely non-zero from the Visibilities directly.

To my knowledge, such an algorithm has never been tried out.

A proof-of-concept algorithm was developed that shows the prediction actually works in practice. It uses Coordinate Descent with the starlet transform, and only needs to calculate a subset of the matrix F^{-1} for a reconstruction. The question remains how many columns of F^{-1} are needed, and is it more efficient than an algorithm using the major cycle architecture.

4 Compressed Sensing with Coordinate Descent and the direct Fourier Transform

Instead of using the Major Cycle architecture and approximate the non-uniform FFT approximation, we directly use the Fourier transform matrix F. We show the principle of our approach and of Compressed Sensing reconstructions in general at a simplified example. Let us minimize the objective function (4.1).

$$\underset{x}{minimize} \ \|V - Fx\|_2^2 + \lambda \|x\|_1 \tag{4.1}$$

The data term $\left\|V - F^{-1}x\right\|_2^2$ forces our image to be similar to the measurements, and the regularization term $\|x\|_1$ tells us how likely the current reconstruction is to be the true image. The parameter λ weights the trade-off between the data and the regularization term. With our term we assume our image contains only a limited number of non-zero pixels. When the instrument observes stars, they take up a single pixel in the image. In this case, our prior models the true image well, and the theory of compressed sensing states we are virtually guaranteed to find the truly observed image at the minimum of our objective function (4.1).

We can use a number of different optimization algorithms to optimize (4.1). Note however in the one dimensional case, where we only have one pixel, the data term of the objective (4.1) forms a parabola and the regularization term a shrink operation[2]. The optimum for a single pixel can be calculated by solving for the minimum of the parabola first, followed by a shrink operation. With Coordinate Descent we can exploit this property. We fix all pixels except for one and iteratively solve for the current minimum of each pixel. Now when we iterate over all pixels several times, we eventually converge to a solution. The full Coordinate Descent algorithm can be implemented in a few lines of python code:

```
def coordinate_descent(V_residual, x, lambda, max_iter):
  for k in range(0, max_iter):
    for i in pixels_row:
      for j in pixels_column:
        x_old = x[i, j]
        fourier_column = calculate_fourier_transform(i, j)
        fr = real(fourier_column)
        fi = imag(fourier_column)
        rr = real(V_residual)
        ri = imag(V_residual)

        #find apex
        a = sum(fr**2 + 2*fr*fi + fi**2)
        b = sum(fr*rr + fr*ri + fi*rr + fi*ri)
        x_new = b / a + x_old

        x_new = shrink(x_new, lambda)
        x[i, j] = x_new
        V_residual = V_residual - fourier_column * (x_new - x_old)
```

Coordinate Descent has to iterate over every pixels possibly several times. How quickly Coordinate Descent converges in theory is not well understood. Our optimization function (4.1) falls in the class of quadratic programming with a strictly convex objective. In this case, Coordinate Descent is guaranteed to converge at least linearly[20]. Sadly, real-world objective functions for image reconstruction are often not strictly convex. Usually our image is constrained to have only positive pixels[9], which breaks the strictly convex property of

[2]The shrink operation reduces the magnitude of the pixel by λ. For example: Pixel $x = -12.4$, $\lambda = 0.6$. The new pixel value after shrinkage follows as $shrink(-12.4, 0.6) = -11.9$

the objective function. In our environment, the convergence guarantees of Coordinate Descent are not well understood in theory.

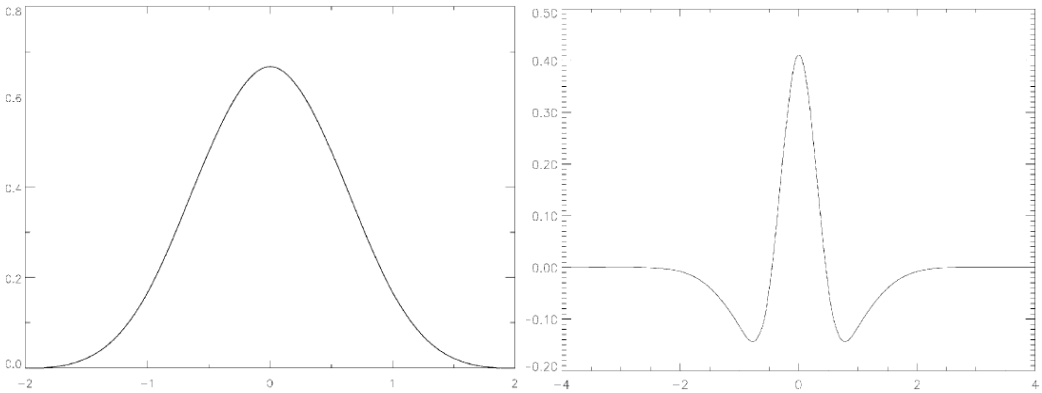
In practice, Coordinate Descent can be improved with heuristics. Since we assume our image contains only a few stars, a few non-zero pixels, the simple Coordinate Descent algorithm wastes resources checking if the pixel is still zero. A better scheme would be the active set heuristic[21]: We iterate over every pixel once. Then, for a given number of iterations, we only check the pixels that have changed. This is an improvement, but too expensive in the context of MeerKAT. A reconstruction problem can have billions of Visibilities and serveral millions of pixels. Since F has the size of Visibilities times pixels, it gets too expensive to calculate all columns even once.

However, it turns out we do not need to if we imitate CLEAN: For CLEAN, the non-uniform FFT approximates the 'dirty' image, which is corrupted by the effects of incomplete measurements. CLEAN then subtracts a fraction of the PSF at the largest pixel value. In other words, the largest pixel value in the dirty image is the most likely non-zero pixel. We can use the dirty image approximation for a 'probability distribution' of non-zero pixels. It does not need to be accurate, since the actual reconstruction is done with the direct Fourier Transform.

This is, in principle, how the proof-of-concept algorithm works. Instead of reconstructing an image, it reconstructs in the starlet space. It uses the starlet transform of the dirty image to find likely non-zero components. Coordinate Descent is used to optimize single starlets. A proof-of-concept version of this algorithm was developed in python.

Although the algorithm produces super-resolved images in section 5, it is currently not known if it actually converges to the true optimum. Finding all relevant non-zero starlets is left to a greedy heuristic. There may be conditions under which it never includes relevant components. Further convergence analysis was outside the scope of the project. Also note that for this implementation, every time likely non-zero components are searched, it simply uses the non-uniform FFT for the dirty image approximation. This was done for simplicity's sake and can be improved.

4.1 The Starlet Transform



(a) Spline scaling function *(b) Wavelet function*

Figure 4: Scaling and wavelet function of the starlet wavelet. Source: [19]

The starlet transform is an Isotropic Undecimated Wavelet Transform (IUWT) with the starlet wavelet shown in figure 4. It defines two operations: The transformation from starlet into image space which is called synthesis, and the inverse which is called decomposition. The starlet space represents an image in multiple layers at different scales, where lower layers contain smaller objects and upper layers the extended emissions. The

lowest starlet layer represents the stars, while the upper layer represents the largest hydrogen clouds in the image.

$$\begin{aligned} c_0 &= \boldsymbol{x} \star B_0 \\ \boldsymbol{w_0} &= \boldsymbol{x} - c_0 \end{aligned} \tag{4.2}$$

Let us look at the lowest starlet layer first, how it is calculated from the image, and how we use it for image reconstruction. The starlet layer w_0 gets calculated in two steps, shown in (4.2). First, we take the scaling function(image 4a) denoted ad B_0, and convolve the image with it. The resulting convolved image is denoted as c_0. The final step is subtracting c_0 from x. This removes larger structures from the image and the starlet layer w_0 is left with only point sources of x.

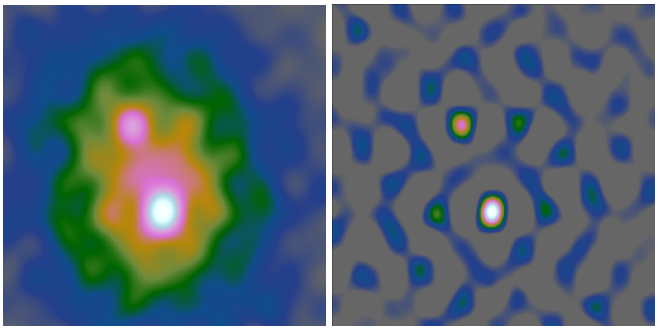
The layer w_0 contains

w_0 has the same dimensions as the input image. It contains the point sources of the image convolved with the starlet wavelet 4b. The last step for finding the point sources α_0, is to deconvolve w_0.

w_0 is not sparse. α_0 is sparse.

Coordinate descent uses the w_0 to see what α_0 is likely to be non-zero.

This is difficult. However we can approximate it with a shrink operation. FISTA scheme of [5]. We use the soft thresholding operation as a probability distribution. We take the w_0 image and see likely relevant pixels and use Coordinate Descent to find its optimum value. A visual example 5.



(a) Dirty Image with two point sources. *(b) Layer w_0 of the starlet decomposition.*

Figure 5: Dirty map and w_0 starlet layer after thresholding. w_0 can be interpreted as a probability distribution for point source locations.

The individual pixels are not independent. The image 5b has a high probability for an area of pixels, although there are only two point sources in the image.

Starlet coefficient α_0 convolved with the starlet wavelet gets back to w_0

On larger scale, we still look at pixels and set a starlet wavelet at the location and scale.

The scaling function grows exponential in size. Each layer is responsible for ever larger structures.

The decomposition in total is chaining the blurring of the scaling function B_i at different scales. Results in the decomposition shown in equation (4.3)

$$\begin{aligned} c_0 &= \boldsymbol{x} \star B_0 & c_1 &= c_0 \star B_1 & & & c_{J-1} &= c_{J-2} \star B_{J-1} & \boldsymbol{c_J} &= c_{J-1} \star B_J \\ \boldsymbol{w_0} &= \boldsymbol{x} - c_0 & \boldsymbol{w_1} &= c_0 - c_1 & \dots & & \boldsymbol{w_{J-1}} &= c_{J-2} - c_{J-1} & & \end{aligned} \tag{4.3}$$

The synthesis transform however is easy. We simply add the starlet layers back together (4.4).

$$\boldsymbol{x = w_0 + w_1 + \ldots + w_{J-1} + c_J} \tag{4.4}$$

The starlet decomposition is over-complete dictionary.

We can use the decomposition as a heuristic, and we do not need to use starlet as reconstruction wavelet. We could also use for example Gaussian functions, and use the starlet decomposition to find out likely locations of what size of Gaussian.

In this work, we did not simply use the starlet wavelet for reconstruction, we modified it to be non-negative.

4.1.1 Handling the non-negativity constraint

Reconstruction algorithms for radio astronomy typically constrain the image to be non-negative[9]. This work orientates itself after the SASIR algorithm[5]. It used the starlet transform together with the FISTA algorithm. Handling the non-negativity constraint in this setting is easier, and we have a translation issue in the context of Coordinate Descent.

The SASIR algorithm decomposes the image into starlet layers as in equation (4.3). It never needs to calculate α_i. Instead, it uses the shrink operation on the individual starlet layers w_i, essentially using it as a constraint on pixels. In this setting, the shrink operator can simply set every negative pixel of w_i to zero. The reconstructed image results from the addition of the individual layers as shown in equation 4.4, x is also non-negative.

Coordinate Descent reconstructs α_i. α_i is constrained to be non negative. However, since we convolve α_i with the starlet, which has negative sections, the resulting w_i also has negative pixels. We have three ways to resolve this:

1. cutting the image x
2. cutting the starlet layers w_i
3. cutting the starlet wavelet

4.2 Coordinate Descent Implementation

We have the starlet regularization, let us put it into an algorithm. Start with the

$$\underset{\alpha}{minimize}\ \|V - FD\alpha\|_2^2 + \lambda \|\alpha\|_1 \tag{4.5}$$

Proof of concept implementation. There are many ways to improve the efficiency and the convergence speed. It is here to show the general if it is possible of the approach and if we can truly only calculate a subset of F^{-1} columns.

The objective function (4.5) minimizes the starlet components α. This objective leads again to a parabola in the one dimensional case, and we can analytically find the optimum for a single component when all others are fixed.

There are still a few problems to solve, like how to handle the matrix product $F^{-1}D$, With the active set heuristic, we iterate over a limited number of starlet components at a time.

The question remains, how the matrix product $F^{-1}D$ can be calculated efficiently. Since starlet transform is just made up of a couple of convolutions, we do not have to calculate the product $F^{-1}D$ explicitly. Convolutions in image space are multiplications in Fourier space. So for any starlet layer w_i or c_J, we can pre-calculate a transform vector. $w_2 = F^{-1}V * (\hat{S}_0\hat{S}_1 - \hat{S}_0\hat{S}_1\hat{S}_2)$

How many layers are needed. starlets rise in pixels with 2^J. Meaning if we want starlets over the whole image, the number of starlets rise logarithmically to the image dimensions.

How to iterate over the image in detail.

4.3 Iteration scheme

Many ways to iterate over the image. A simple scheme was used here.

Active set heuristic, find the starlets which are

```
def coordinate_descent(V_residual, x, lambda, max_iter):
  for k in range(0, max_iter):
    for i in pixels_row:
      for j in pixels_column:
        x_old = x[i, j]
        fourier_column = calculate_fourier_transform(i, j)
        fr = real(fourier_column)
        fi = imag(fourier_column)
        rr = real(V_residual)
        ri = imag(V_residual)

        #find apex
        a = sum(fr**2 + 2*fr*fi + fi**2)
        b = sum(fr*rr + fr*ri + fi*rr + fi*ri)
        x_new = b / a + x_old

        x_new = shrink(x_new, lambda)
        x[i, j] = x_new
        V_residual = V_residual - fourier_column * (x_new - x_old)
```

coordinate descent with active set heuristic

sub-zero pixels

4.3.1 Non-uniform FFT approximations

Non uniform FFT to calculate the convolution KERNELS!!

Stupid approach with line search. Could be done more efficiently by using the histogram of the starlet level.

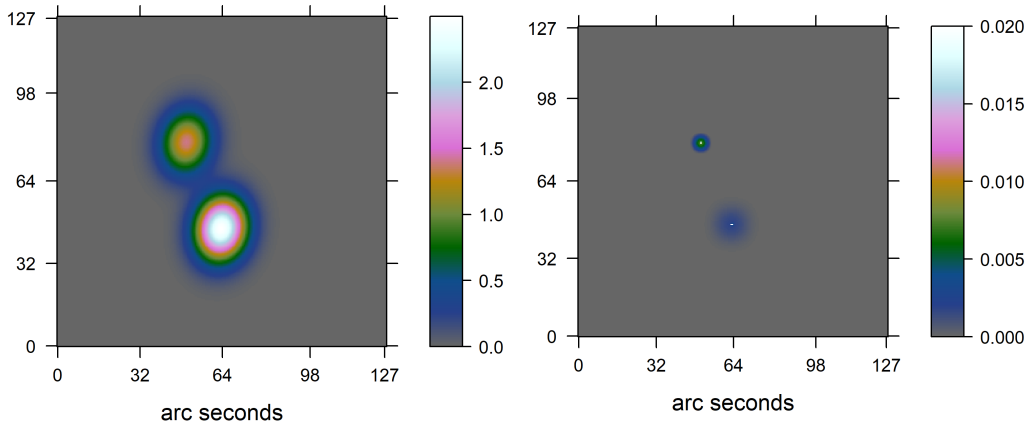
5 Test on simulated data

In this section, we test our new approach with Coordinate Descent on simulated MeerKAT data. We show that our approach does not need to calculate the whole Fourier Transform matrix F. Instead, we use a heuristic to only use relevant columns. As mentioned in section 4, it is unknown if our approach will converge to the true optimum. Nevertheless, we compare our results with CASA's CLEAN implementation, and demonstrate super-resolution performance of Coordinate descent together with accurate total flux modelling.

The two simulated datasets contains idealized MeerKAT observations. Compared to the real world, the two simulated datasets contain few Visibilities and not representative of the real data volume. Also, more realistic simulations which contain pointing-, calibration-, and thermal noise are out of scope for this project. The simulations are used to isolate the two fundamental issues in radio interferometer image reconstruction: Non-uniform sampling and incomplete measurements.

5.1 Super-resolution of two point sources

The first simulated observation contains two non-zero pixels, i.e. point sources, with intensity of 2.5 and 1.4 Jansky/Beam. The image has a size of 256^2 at a resolution of 0.5 arc-seconds per pixel. The integral, the total Flux of the image, is 3.9 Jansky/beam.



(a) CLEAN reconstruction with CASA standard parameters.

(b) Coordinate Descent reconstruction with $\lambda = 0.01, J = 4$.

Figure 6: Image reconstruction of two simulated point sources.

The figure 6 shows the CLEAN and the Coordinate Descent reconstruction. CLEAN reconstructs the image 6a at the accuracy limit of the instrument. It essentially reconstructs a blurred version of the observed image, where the blurring represents the accuracy of the instrument. With compressed sensing, we aim to reconstruct the de-blurred image, increasing the effective accuracy of the instrument.

Coordinate Descent in image 6b shows a super-resolved reconstruction of the two point sources. It reconstructs two narrow peaks surrounded,by a low-intensity Gaussian emission. Also, Coordinate Descent manages to capture the total flux more accurately than CLEAN. The total flux of image 6b results in 3.92 Jansky/beam, compared to CLEAN which overshoots the number by a factor of 1400. The total flux of the CLEAN reconstruction 6a overshoots the 3.9 figure by a factor of 1400. This gets obvious when we compare the intensity profile of CLEAN, Coordinate Descent and the ground truth in figure 7.

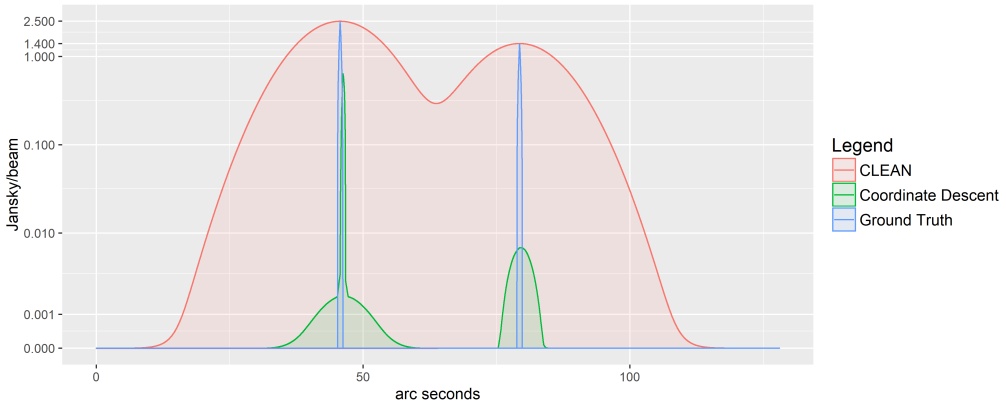
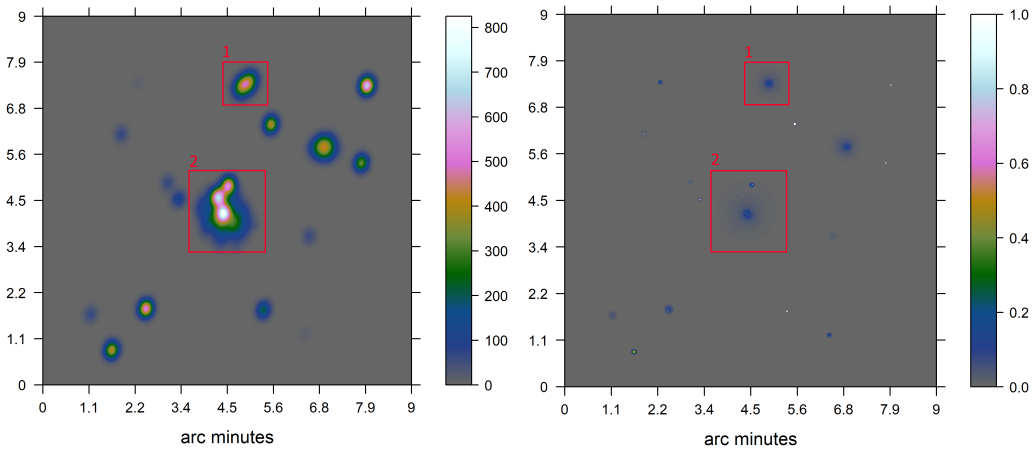


Figure 7: Intensity profile of the two point sources.

CLEAN essentially places a Gaussian function with correct peak intensity at the point source location, but it does not respect the total flux of the image. Coordinate Descent keeps the total flux in mind. The pixels in each area of the point sources sum up to the correct values of 2.5 and 1.4 respectively. However, note that Coordinate Descent in figure 7 seems to have both point sources shifted by approximately a pixel. It looks suspiciously like an off-by one error. Sadly in the time frame of this project, no error was found or an explanation for this behaviour.

5.2 Super resolution of mixed sources

This dataset contains a mixture of three Gaussian emissions and sixteen point sources of varying intensities. At the center of the image, it has three point sources underlying a weak extended emission. The image center and one Gaussian emissions are analysed in detail. Coordinate Descent was run for two full iterations, using $\lambda = 0.01$ and $J = 7$ starlet layers. The CLEAN and Coordinate Descent reconstructions are shown in figure 8. Our algorithm was able to locate the point sources below the accuracy limit of MeerKAT. However, some point sources were reconstructed with artefacts.



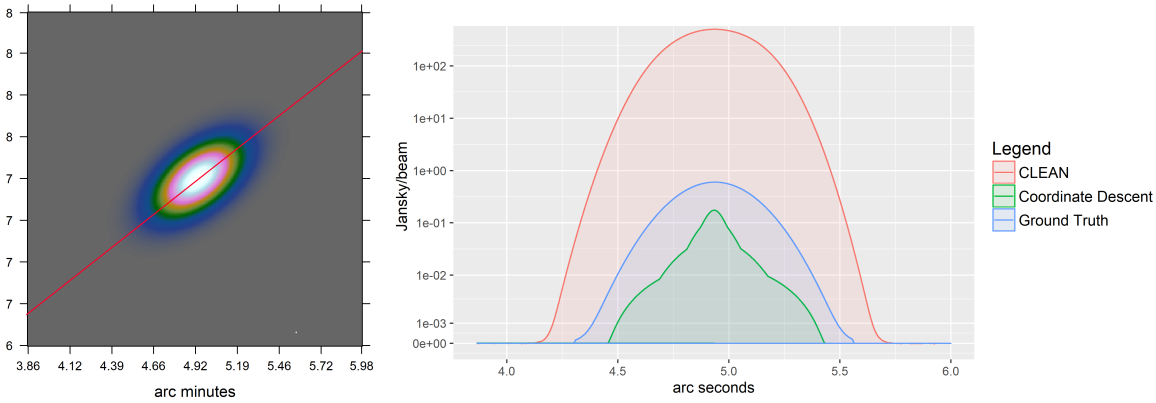
(a) CLEAN reconstruction *(b) Coordinate Descent Reconstruction*

Figure 8: Reconstruction on mixed sources

Look at the fucking

The number of Fourier columns scales with the number of non-zero components

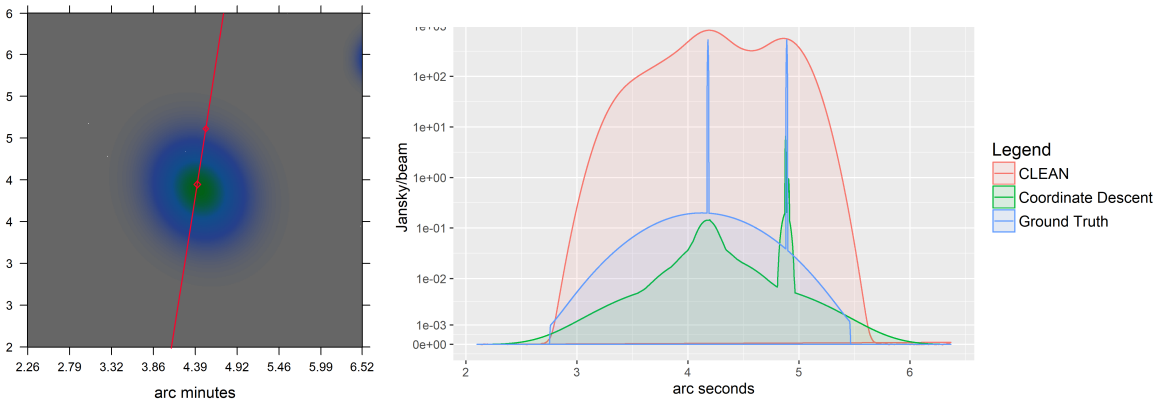
Question of Flux reconstruction



(a) Ground truth. *(b) Intensity profile.*

Figure 9: Intensity profile of region 1.

More accurate total flux reconstruction, but no



(a) ground truth *(b) ground truth*

Figure 10: Intensity profile of region 2.

More complex source. We see a super resolution and again more accurate flux. Point sources on the other hand, do become problematic. We see one point souce split up in two very narrow peak. When we look at other point sources of figure 11, we see mixed results. Some point sources are reconstructed as we would wish, with one high peak in the center, locating the point source, and a small starlet artifact which is the low intensity ”glow” around it. The first point source is what we would wish for. We show a few failure modes of point source reconstruction. A lower intensity ”trail” in the second image and a split into two point sources of images three and four.

problem of an error, since Coordinate Descent should have converged on the two point sources.

Two things could be made to improve it: This project uses the same λ for all starlet layers. Girard et al.[5] used a different λ for each layer.

different way of handling positivity.

The trail issue of point sources can be an artifact of how positivity is handled in our algorithm.

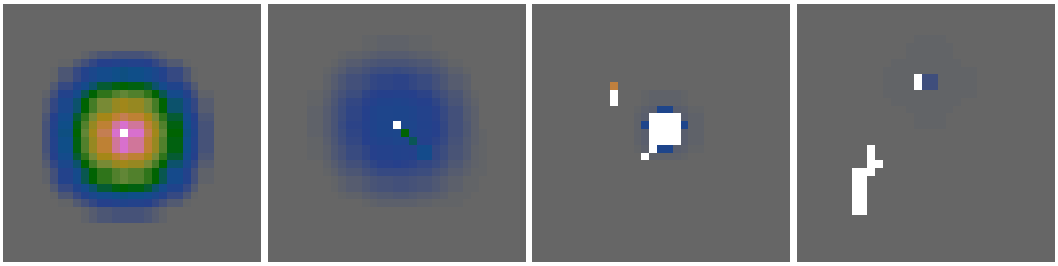


Figure 11: Different point source reconstructions of Coordinate Descent.

Starlet reconstruction with negative parts, but leads to a solution with far more non-zero components, [since it has to counter-act the negativity]. The reconstruction 8b is the result of 244 non-zero, strictly non-negative starlets. Letting the starlet have its negative 'dip' results in a reconstruction with more than ten times the components.

244 non-zero starlets,

show the areas

6 Runtime cost comparison on a real-world MeerKAT reconstruction

Current Compressed Sensing reconstructions produce images at a higher quality than CLEAN. However, CLEAN is significantly cheaper to compute. MeerKAT's large scale reconstruction problems, CLEAN is still the go-to algorithm. In this project, we developed for a new architecture, which uses the relevant columns of the Fourier Transform Matrix directly. We developed a Coordinate Descent algorithm with this architecture and demonstrated in section 5 super-resolution performance on simulated data. The question, if we can lower the runtime costs with our new architecture, is still open.

In this section, we compare the costs of Coordinate Descent with WSCLEAN, which is the reconstruction algorithm of choice for MeerKAT reconstructions. We create cost functions for each algorithm, which estimate the number of operations depending on the input size. WSCLEAN was executed on a real-world MeerKAT dataset shown in image 12. Our proof-of-concept implementation was not able to handle the large amount of data. Instead, we extrapolate the best-case costs of our approach and compare them to WSCLEAN on the MeerKAT dataset.

6.1 Cost function of an idealized Coordinate Descent

The runtime cost of Coordinate Descent depends on the number of Visibilities M and the number of non-zero starlets S. The number and location of the S non-zero starlets are generally not known. However, we created a heuristic which finds likely non-zero starlet components. In a realistic setting, the heuristic will have found more than S likely non-zero starlets. For the idealized version of Coordinate Descent, we assume an oracle performance heuristic: It finds the location and number of the S non-zero starlet components in constant time. Coordinate Descent therefore has to calculate the value of S components. In total, the idealized Coordinate Descent algorithm uses four steps: creating J starlet levels with the non-uniform FFT, creating the columns of F^{-1}, calculating the minima for each single component, and calculating the starlet layers:

$$\begin{aligned} J\text{non-uniform FFTs for the starlet regularization} &: J * (M + 2N * ld(2N)) \\ \text{creating } S \text{ columns of } F^{-1} &: S * 7M \\ \text{locating } S \text{ minima of } S \text{ parabolas} &: S * 4M \\ \text{calculating } J \text{ Starlet layers} &: J * 2M \end{aligned}$$

We assume we have enough memory to cache the columns of F^{-1} and only need to calculate them once. Keep in mind that each column of F^1 has the same length as the Visibilities, essentially multiplying the input data. The last parameter for Coordinate Descent is the number of iterations to converge, $I_C D$. Estimating this number is difficult as Coordinate Descent does not have strict guarantees (as discussed in section 4). Instead, we assume it converges after a fixed number of iterations. Therefore we arrive at the cost function of (6.1).

$$\begin{aligned} CD(I_{CD}, M, S, J) = & I_{CD} * [S * 4M + J * 2M] \\ & + S * 7M \\ & + J * (M + 2N * ld(2N)) \end{aligned} \tag{6.1}$$

Note that the runtime of Coordinate Descent is independent of the number of pixels. The only image related parameter in (6.1) is J, the number of starlet layers. The largest starlet layer represents the largest possible structure in the image, which is given by the instrument and the image resolution. The runtime only depends indirectly on the image resolution, not the total number of pixels. For simplicity, we assume the image cannot

have structures larger than half the image size. For our MeerKAT example, this is more than enough to represent the largest structures.

Also note the term iterating over the S non-zero starlets, $I_{CD} * [S * 4M + \ldots]$. As it turns out, this is the Achilles heel of the algorithm. MeerKAT observations contain a very large amount of Visibilities M.

6.2 Cost function of WSCLEAN

The WSCLEAN algorithm uses the Major Cycle architecture. It uses the non-uniform FFT with w-stacking. The runtime costs of a single Major Cycle depends on the non-uniform FFT with w-stacking and the number of CLEAN deconvolutions. N denotes the number of pixels.

$$\begin{aligned} \text{non-uniform FFT} &: M + 2N * ld(2N) \\ \text{non-uniform FFT with } w\text{-stacking} &: M + W * (2N * ld(2N) + 2N) + N * ld(N) \\ I_{WSCLEAN} \text{ deconvolutions} &: I_{WSCLEAN} * 2N \end{aligned}$$

The overall cost function shown in (6.2) can also be split into two parts. In each Major Cycle, the forward and backwards non-uniform FFTs gets calculated, and CLEAN deconvolves the image for a certain number of iterations.

$$\begin{aligned} WSCLEAN(I_{Major}, I_{CLEAN}, M, N, W) = I_{Major} * 2 * [M + W * (2N * ld(2N) + 2N) + NlogN] \\ + I_{Major} * [I_{CLEAN} * 2N] \end{aligned} \tag{6.2}$$

Notice that the number of CLEAN deconvolutions I_{CLEAN} depends on the image content, similar the number of non-zero starlets S for Coordinate Descent. Here however, it multiplies with the number of pixels instead of the number of Visibilities. In a sense, the major cycle tries to reduce the runtime complexity of handling the image content by calculating the non-uniform FFT. If the difference is large enough $N \ll M$, then the Major Cycle will end up with a smaller runtime costs.

6.3 Comparison on a MeerKAT reconstruction problem

Our real-world MeerKAT observation has been calibrated and averaged in frequency and time to reduce storage space. The resulting dataset contains 540 channels with 4 million Visibilities each. Due to hardware limitations, WSCLEAN was calculated on 75 channels. The reconstructed image is shown in 12, and the resulting parameters for our cost function are:

- Major Cycles: $I_{Major} = 6$
- Number of CLEAN iterations: $I_{CLEAN} = 35'000$
- Visibilities: $M = 3.05e^8$
- Pixels: $N = 2048^2$
- w-stacks: $W = 32$

For Coordinate Descent's costs, we need an estimate for J, S and I_{CD}. We set $J = 8$, which lets the largest structure span over half the image, enough to capture any large scale structures. For S and I_{CD}, we simply

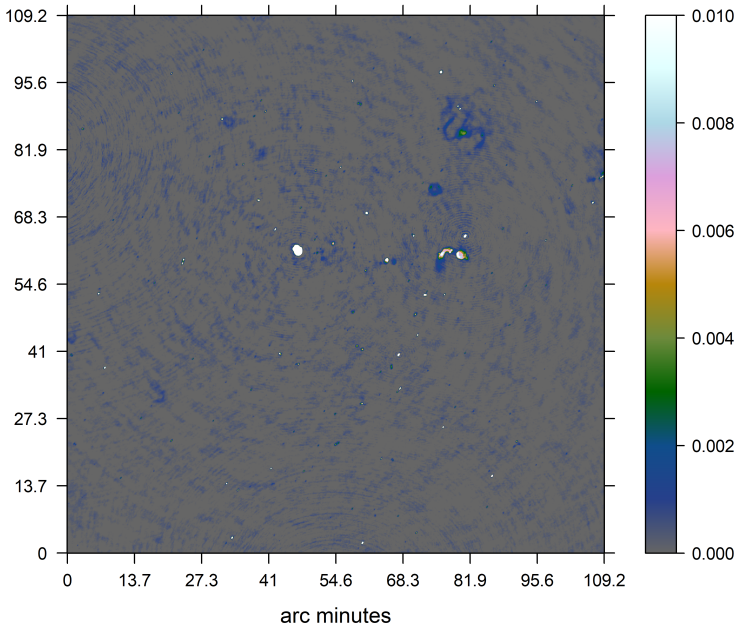


Figure 12: WSCLEAN Reconstruction of the MeerKAT observation.

use the values from our simulated reconstruction 8b and set $S = 250$ and $I_{CD} = 10$. This is an underestimation of the true values. The image 12 shows complex-shaped extended emissions, which likely needs a larger number of starlets for representation than the Gaussian emissions from our simulation.

When we put all values into our cost functions (6.1) and (6.2), Coordinate Descent with the direct Fourier transform arrives at 1.9 times the costs of WSCLEAN in the best case scenario. Our approach has not reduced the runtime costs compared to WSCLEAN. But what about Major Cycle Compressed Sensing algorithms? We have not developed a cost function for other Major Cycle approaches, but we can get a very rough estimate by changing I_{Major} of the WSCLEAN cost function. Pratley et al.[10] reported 10 Major Cycles for a Compressed Sensing reconstruction on simulated data. If we plug in 10 Major Cycles for WSCLEAN, Coordinate Descent ends up taking roughly 1.2 times the cost of the Major Cycle algorithm. Mind you these costs are only valid when we can keep all necessary columns of F in memory, eating up 1.1 terabytes[3] in our case. If not, the columns have to be re-calculated on the fly, increasing the runtime costs by factors.

The issue with Coordinate Descent's runtime complexity lies in the term $I_{CD} * [S * 4M + \ldots]$ of (6.1), which scales with the "content" of the image S, multiplied with the Visibilities. Coordinate Descent cannot afford many iterations nor many non-zero components, because both of these numbers get multiplied together with M, the largest number in the problem. With the Major Cycle architecture, WSCLEAN is able to get around this limitation, and scales any content dependent factors on N instead of M.

Indeed, the runtime of our Coordinate Descent algorithm could be improved by using the major cycle architecture, essentially replacing M with N and we arrive at the term $I_{CD} * [S * 4N + \ldots]$. By using the Major Cycle architecture, Coordinate Descent can afford more iterations and more non-zero components in the image for the same runtime complexity. Furthermore N lies on a uniformly sampled grid. We may be able to use the FFT instead of caching columns of F, and reduce the memory requirement at the same time.

[3] If we assume 64 bit floating point accuracy for the real and complex values of the Visibilities.

6.4 Approximations as key for going large scale

In this project, we created a new architecture for Compressed Sensing reconstructions. Instead of using the non-uniform FFT approximation, we directly use the relevant columns of the Fourier transform matrix. We showed that the likely-relevant columns can be retrieved with a heuristic, and we can reconstruct images without ever calculating the whole Fourier matrix. This architecture naturally extends to the wide field of view measurement equation (2.1). It can handle the w-term explicitly and does not need any approximations.

We created a proof-of-concept algorithm in this architecture, with Coordinate Descent as the optimizer and starlets as regularization. We exploited the starlet for a greedy heuristic, locating the likely-relevant columns of the Fourier transform matrix. The same heuristic can be used with other regularizations. It locates the likely location of Gaussian-like objects at different scales. As such the same heuristic could be used for a Gaussian mixture model. At this point, it is unclear if the heuristic will eventually converge to the true optimum in theory. In practice, we showed super-resolution performance of Coordinate Descent on simulated MeerKAT data together with accurate total flux modelling compared to CLEAN.

This is a similar result to other Major Cycle compressed sensing algorithms, which have also shown super-resolved reconstructions compared to CLEAN on real-world observations[6][5], but also at a higher runtime cost. The main goal of this project was to find a new architecture which reduces the runtime cost of Compressed Sensing algorithms for large scale reconstructions. In the context of MeerKAT, the estimated runtime costs of CLEAN were still half of the best case estimate of our Coordinate Descent algorithm. Ironically, the runtime costs of Coordinate Descent can be improved by using the Major Cycle architecture instead of the direct Fourier transform and potentially reduce its memory requirement as well.

That does not mean the new architecture cannot beat CLEAN or the Major Cycle. Our architecture scales independently of the image size. The cost effectiveness of the two architectures boils down to the ratio between Visibilities and pixels. For MeerKAT reconstructions, the ratio is skewed towards Visibilities, and the Major Cycle is more cost effective. But if we increase the number of pixels in the reconstruction, we increase the cost-effectiveness of our direct Fourier transform architecture.

Increasing the number of pixels is only useful to a limit. Decreasing the number of Visibilities however, might yield better results: The large number of Visibilities are virtually guaranteed to have redundant information. A portion of the total runtime is spent on Visibilities, which do not contain any new information for the image. One improvement may be to sub-sample the Visibilities and reconstruct an approximate image from a fraction of the dataset. In this environment, our new architecture may become viable for reducing runtime costs. Indeed, the direct Fourier transform may not be cost effective partly because it does not use approximations. Our architecture calculates the Fourier transform up to the limit of floating-point precision. The Major Cycle on the other hand, uses the non-uniform FFT approximation and stops at the noise limit of the measurements. For our new architecture, introducing back approximations may be the key to reduce runtime costs.

In the future MeerKAT is scheduled to be expanded, posing the reconstruction problem on an even larger scale. Sooner or later, the reconstruction has to be done over a distributed computing infrastructure. In the Major Cycle, single steps like the w-correction lend themselves to distributed computing. But designing a Major Cycle algorithm which is highly distributable is an open problem. The simplified architecture of our direct Fourier transform together with Coordinate Descent may have an edge in distributed computing. Coordinate Descent can converge, even when the descent steps are distributed. As it exists in this project, the memory and runtime costs of our approach are too high to be relevant for MeerKAT reconstructions.

However, if we can decrease the memory and runtime costs with the right approximations, while keeping the distribution properties of Coordinate Descent intact, our approach may become interesting for distributed image reconstructions.

7 Compressed Sensing reconstructions in the MeerKAT era

MeerKAT poses a large scale reconstruction problem. Compressed Sensing approaches exist, they use the major cycle, but they all are slower to compute than CLEAN, for which the major cycle was developed.

It uses the non-uniform FFT to approximate the Fourier transformation of non-uniformly sampled Visibilities.

This project set out to explore different architectures for Compressed Sensing Reconstructions. A different way of getting from Visibilities to image.

Three different possibilities, with calculating the non-uniform FFT once, using spherical harmonics, or calculating only columns of the Fourier transform matrix

For the former, we created a novel approach which does not need the major cycle. We need a transformation, where we can both estimate which components are likely to be non-zero, and which use few non-zero components to represent an image.

This project used starlets, which were developed for astronomical images. Their solution is not sparse enough to scale to MeerKAT problems.

This leaves two other ways, splitting the major cycle falls off if we consider self-calibration

Spherical harmonics are the last way, Compressed Sensing algorithms have used it to gain another improvement in reconstruction quality. At this point, there is no Compressed Sensing algorithm which used the sphere for speed up.

here I see a possibility. The downside is it leads to a new measurement equation. It did not have as much time to formalize corrupting effects like antenna beam pattern, ionosphere and so on.

There is yet an

Still an open question

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] JW Rich, WJG De Blok, TJ Cornwell, Elias Brinks, Fabian Walter, Ioannis Bagetakos, and RC Kennicutt Jr. Multi-scale clean: A comparison of its performance against classical clean on galaxies using things. The Astronomical Journal, 136(6):2897, 2008.

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

[5] 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.

[6] 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.

[7] Jasleen Birdi, Audrey Repetti, and Yves Wiaux. Sparse interferometric stokes imaging under polarization constraint (polarized sara). arXiv preprint arXiv:1801.02417, 2018.

[8] Tim J Cornwell, Kumar Golap, and Sanjay Bhatnagar. The noncoplanar baselines effect in radio interferometry: The w-projection algorithm. IEEE Journal of Selected Topics in Signal Processing, 2(5):647–657, 2008.

[9] 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.

[10] 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.

[11] Luke Pratley, Jason D McEwen, Mayeul d'Avezac, Rafael E Carrillo, Alexandru Onose, and Yves Wiaux. Robust sparse image reconstruction of radio interferometric observations with purify. Monthly Notices of the Royal Astronomical Society, 473(1):1038–1058, 2017.

[12] Stephen J Hardy. Direct deconvolution of radio synthesis images using l1 minimisation. Astronomy & Astrophysics, 557:A134, 2013.

[13] Arwa Dabbech, Laura Wolz, Luke Pratley, Jason D McEwen, and Yves Wiaux. The w-effect in interferometric imaging: from a fast sparse measurement operator to superresolution. Monthly Notices of the Royal Astronomical Society, 471(4):4300–4313, 2017.

[14] TD Carozzi. Imaging on a sphere with interferometers: the spherical wave harmonic transform. Monthly Notices of the Royal Astronomical Society: Letters, 451(1):L6–L10, 2015.

[15] JD McEwen and AMM Scaife. Simulating full-sky interferometric observations. Monthly Notices of the Royal Astronomical Society, 389(3):1163–1178, 2008.

[16] Nathanaël Schaeffer. Efficient spherical harmonic transforms aimed at pseudospectral numerical simulations. Geochemistry, Geophysics, Geosystems, 14(3):751–758, 2013.

[17] J Richard Shaw, Kris Sigurdson, Ue-Li Pen, Albert Stebbins, and Michael Sitwell. All-sky interferometry with spherical harmonic transit telescopes. The Astrophysical Journal, 781(2):57, 2014.

[18] Jason D McEwen, Gilles Puy, Jean-Philippe Thiran, Pierre Vandergheynst, Dimitri Van De Ville, and Yves Wiaux. Sparse image reconstruction on the sphere: implications of a new sampling theorem. IEEE Transactions on image processing, 22(6):2275–2285, 2013.

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

[20] Zhi-Quan Luo and Paul Tseng. On the convergence of the coordinate descent method for convex differentiable minimization. Journal of Optimization Theory and Applications, 72(1):7–35, 1992.

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

List of Figures

List of Tables

8 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, January 27, 2019

Jonas Schwammberger