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

Jonas Schwammberger

January 18, 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[**?**][**?**] 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[**?**][**?**]. 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[**?**][**?**][**?**]. 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

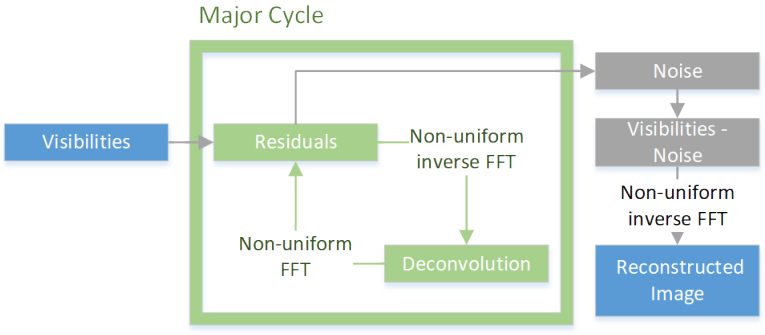


Figure 1: The Major Cycle Framework

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[**?**].

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

The measurement equation of radio interferometers contains a third fourier component, the w component. This leads to the following measurement equation (2.1).

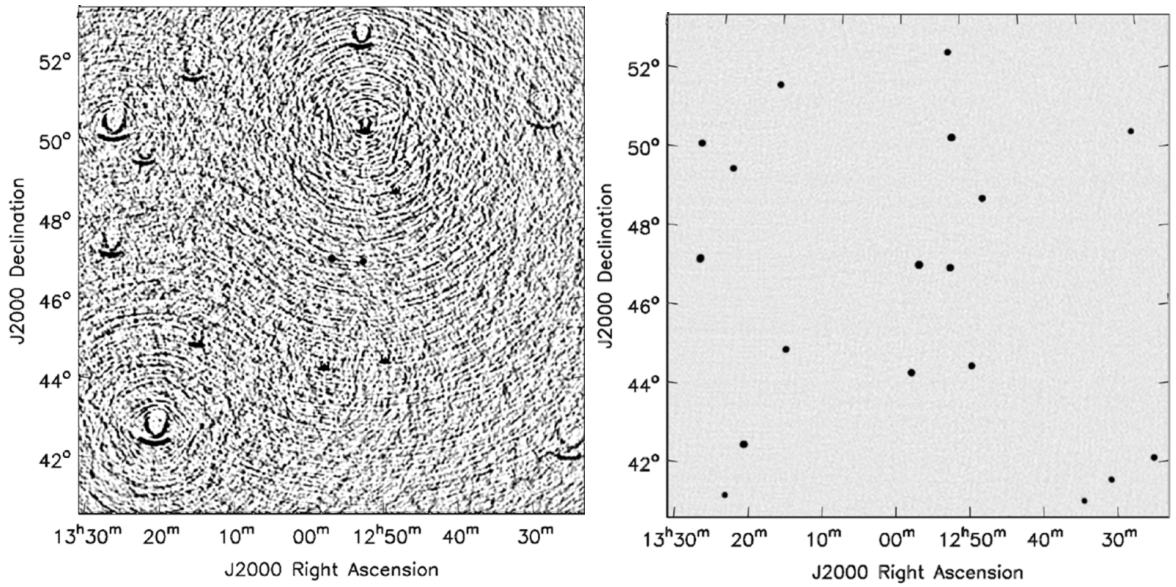
$$V(u,v,w) = \int\int \frac{I(x,y)}{\sqrt{1-x^2-y^2}} e^{2\pi i[ux+vy+w(\sqrt{1-x^2-y^2}-1)]}\, dx\, dy \tag{2.1}$$

For small field of view observations, the term $\sqrt{1-x^2-y^2} \approx 1$, and we arrive at the basic measurement equation (1.1). For small field of view, the w term can be ignored, and the image can be calculated with the inverse two dimensional non-uniform FFT. For wide field of view measurements, the w term breaks the two dimensional Fourier relationship. The measurement equation (2.1) can still be inverted, but the resulting algorithm has a quadratic runtime and is not feasible in practice.

Note that the image $I(x,y)$ of the wide field of view the measurement equation (2.1) is still two dimensional, even though the Visibilities $V(u,v,w)$ have three. The relationship between Visibilities and image is neither the two, nor the three dimensional Fourier transform. The Visibilities represents an image on the celestial

sphere. This means the image is naturally curved, and the w component represents the distortion introduced by the curvature.

One can simply ignore the w component, ignore the curvature and use the two dimensional Fourier Transform. The image then does not lie on a sphere, but on a tangent plane. At the tangent point (where $\sqrt{1-x^2-y^2} \approx 1$, typically the image center in radio astronomy) the two reconstructions are identical. But the distortion gets more severe away from the tangent point. The image 2a shows a tangent plane reconstruction over a large field of view, while 2b shows the same reconstruction but w corrected. Close to the tangent point, the image center, the reconstructions both locate the point sources. At the image edges, the distortion gets severe enough to decorrelate the Visibilities and the point sources get 'torn' apart.



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

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

Image reconstructions for MeerKAT need to account for the w component efficiently. In the Major Cycle framework, this is part of the non-uniform FFT. The first approach was to image facets, creating several tangent planes for the celestial sphere. A more efficient approach came in the form of the w-projection algorithm[**?**]. It uses convolution on the Visibilities during the non-uniform FFT. MeerKAT currently uses the w-stacking algorithm, which more efficient than w-projection on MeerKAT's data volume and is simple to distribute.

This is still an active field of research. The latest work as of time of writing is a hybrid approach[**?**] with both w-stacking and w-projection. As of time of writing, it was not yet adapted.

2.1.1 State of the art: w-stacking algorithm

Here a short explanation of the w-stacking algorithm is provided. It is a surprisingly simple algorithm that seems to be here to stay. It is a modification of the nuFFT operation displayed in figure 1.

Inverse nuFFT with w-stacking: Visibilities get grouped into stacks by their w-term. Every visibility inside a stack has a similar w-term. The more stacks, the more accurately you can handle the w component. The figure displays the relation

The nuFFT gets called on each stack separately, and a constant w correction factor applied. Since w-term is constant, we can use the two dimensional nuFFT again. At the end, all the stacks are summed together to form the dirty image.

nuFFT with w-stacking: The inverse is done in a similar fashion. The sum cannot be re-separated, so the image gets copied into each stack. The constant w factor gets divided out. The nuFFT gets applied.

the runtime

(talking about more recent hybrid approach, limiting the total number of ?w-stacks?)

2.2 Self-Calibration

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

There is little research in exploring a different architecture for Compressed Sensing. The closest to this work was done by Pratley et al[**?**], which optimizes the non-uniform FFT in the context of compressed sensing. The Major cycle is still used. On the upside, progress in w-term accuracy and runtime.

There are ways to form a different architecture. However, the advent of self-calibration pushes a forward-backward kind of architecture on the reconstruction.

3.1 Separating the Major Cycle into two Optimization Problems

The major cycle reduces two errors simultaneously. The error introduced by the non-uniform FFT approximation, and the error introduced by the deconvolution approximation. In a sense, the major cycle is a simple optimization algorithm: It iteratively calls the non-uniform FFT and deconvolution approximations with the error from the last iteration, until the error is below a certain threshold.

$$FT: \underset{I_{dirty}}{minimize} \left\| V - A^{-1} I_{dirty} \right\|_2^2 \tag{3.1}$$

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

The major cycle can also be separated into two objectives (3.1) and (3.2), and solved in sequence. The first objective (3.1) tries to find the best uniformly spaced image I_{dirty} matching the observation. A^{-1} represents the non-uniform inverse FFT approximation ($A^{-1} I_{dirty} \approx F^{-1} I_{dirty}$). When this is solved, the original Visibility measurements are not needed any more. Then, the second objective (3.2) searches for the deconvolved image x regularized by some prior $P()$.

By separating the two objectives, the hope is we can use specialized optimization algorithms that overall converge faster than several major cycles. Also, after the first objective was minimized, we can drop the Visibilities, which can be magnitudes larger than the image.

At first glance, this is a sensible idea, but the devil is in the objective (3.2). Namely in the PSF is not constant, it varies over the image. In the Major Cycle Architecture, a constant PSF[1] is sufficient: It only needs to approximate the deconvolution. The error introduced by a constant PSF can be reduced in the next Major Cycle. By eliminating the Major Cycle, we also eliminated the chance to approximate the deconvolution with a constant PSF. In the worst case, we would need to estimate the PSF for each pixel in the image.

In the worst case, these many PSFs may stop us from using this approach. It is plausible that can use fewer PSFs and be close enough. No known work done in the radio astronomy. The Major Cycle is ubiquitous. But it is plausible.

However, the problem is that the imaging algorithm should also be able to calibrate. With this approach, there is a strict one way flow from measurements to image. For calibration, a backwards flow from image to measurements is necessary.

For calibrated data, this approach may be potentially faster. but since MeerKAT will be difficult to calibrate correctly, this is unlikely.

Fourier Transform for PSF

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

3.2 Reconstruction on the Sphere

The signal naturally lives on the celestial sphere.

Not that much used in Radio Astronomy reconstructions. Work that uses to project on the sphere and looks at the reconstruction quality [**?**].

The wide field of view measurement equation (2.1) can be expanded into spherical harmonics.

[**?**] measurement equation with spherical harmonics. Essentially replaces the non-uniform FFT from the Major cycle with a new approach based on the fact that it has a solution to the Helmholtz equation. , see [**?**] for further detail. It can be used as a different way to arrive at a dirty image from wide field of view interferometers. It is useful in full sky image, because it fulfills the (property from paper) and the full sky dirty image does not have artifacts. It does not gain any complexity advantage, quite the contrary.

Even if the spherical harmonics transform would gain a speed advantage, used in the way of [**?**] it would still be the same major cycle architecture, just with spherical harmonics transform.

However there is the possibility to just do everything on the sphere.

Spherical haar wavelets used for simulating visibilities from a known image, essentially the opposite operation from imaging. [**?**]

based on the spherical harmonic transform. Doing a Spherical harmonic transform efficiently is still an active field of research [**?**]

transform . The wide field of view measurement equation (2.1) can be factorized into spherical harmonics. This means spherical harmonics are equivalent, calculating an image from the measurements with the three dimensional fourier transform is the same as using the spherical harmonics transform. [**?**]

[**?**]sampling theorem on a sphere

Spherical harmonics were researched in the context of Compressed Sensing image reconstructions (cite wiaux). They improved the quality of the image reconstruction, but no speed advantage.

The push on solving on the sphere directly. Using spherical haar wavelets, (cite) was able to speed up the simulation problem. Simulation tries to solve to problem of finding the measurements corresponding to a given image x. However, this has so far not been used for imaging.

3.3 Explicit Fourier Transform Matrix

Hardy et al[**?**] 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 (4gb of visibilities times 1 Billion Pixels).

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 image. The starlet transform[**?**] 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

The three components of a compressed sensing reconstruction. Using a combination so we do not need to calculate the whole Fourier Transform matrix. Regularization represents our previous knowledge of possible images. With it we find the most likely image which represents our data.

Doing compressed Sensing with Coordinate Descent and starlet regularization.

The idea behind compressed sensing is we use prior information about the image, and find the most likely image x from all possible solutions. In practice, we formulate a minimization problem with a data and regularization term. The data term forces the image to be as close to the measured Visibilities as possible, and the regularization term tells us how likely the image is to be true.

The regularization is responsible for an accurate reconstruction. In this work, two regularizations are used. The L1 pixel regularization is used to demonstrate a simple version of the coordinate descent algorithm. For the actual algorithm starlet regularization is used. Starlets have been used as regularization for the LOFAR interferometer[**?**], which was able to produce super-resolved reconstructions.

First, let us look at the coordinate descent algorithm in its simplest form, and discuss its properties. we want to minimize the objective (4.1), where the data term $\left\|V - F^{-1}x\right\|_2^2$ forces the image to be close to the Visibilities V, and the regularization term $\|x\|_1$ forces the image to have as few non-zero pixels as possible. The parameter λ represents our trade-off between reconstruction accuracy and regularization.

$$\underset{x}{minimize} \ \left\|V - F^{-1}x\right\|_2^2 + \lambda \|x\|_1 \tag{4.1}$$

The objective function (4.1) has a property, which we can exploit with Coordinate Descent: The optimum for a single pixel, if we keep all others fixed, turns out to be a parabola. The optimum for a single pixel of (4.1) is simply finding the apex of a parabola, followed by a shrink operation[2]. Sadly, the pixels are not independent of each other, and we need multiple iterations over all pixels to converge. This leads to the following reconstruction algorithm, written in 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)
```

[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 simple version of coordinate descent converges to a solution which is similar to CLEAN. The convergence rate of coordinate descent is not well understood. Our image reconstruction problem falls in the class of quadratic programming, in coordinate descent converges at least linearly[**?**]. (Strict convex function, may not be true in this section. This a lose bound given by the theory). In practice, the convergence rate can be sped up with heuristics like active set[**?**] and λ[**?**]. Furthermore, coordinate descent is robust, even when the residual vector is only approximated, which is useful for distributing the algorithm.

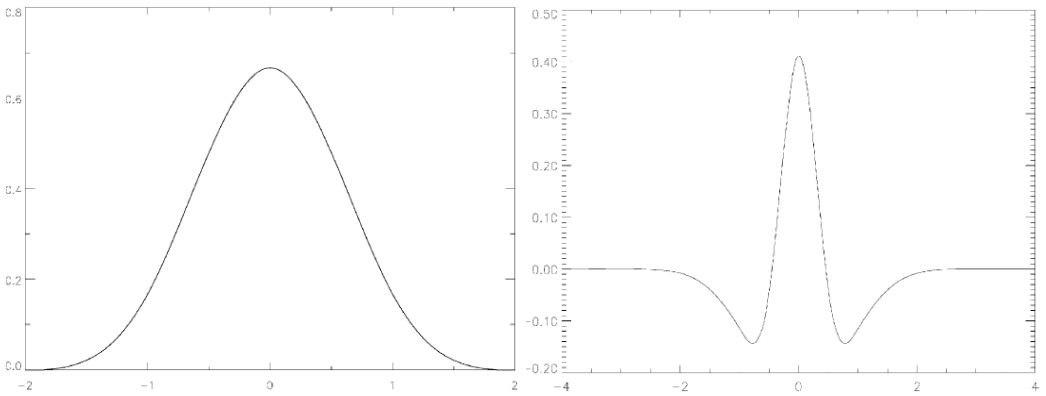
The simple approach calculates the whole Fourier Transform Matrix F^{-1} explicitly. The upside is that we do an exact w-term correction. The downside is F^{-1} is too large to keep in memory, and too expensive to calculate on the fly for the scale of MeerKAT reconstructions.

The starlet transform is used from here on. Instead of reconstructing the image directly, we represent the image as a combination of starlets $x = D\alpha$. Where D is an over-complete dictionary of starlets, and α is the starlet component vector. Since the dictionary D is over-complete, it is a matrix with more columns than rows, and does generally not have an inverse (more formally: $D^{-1}x \neq \alpha$). However, the starlet transform does have an pseudo-inverse. In this work, the pseudo inverse is used to only look at a subset of starlet components which are likely to be non-zero, and only evaluate a subset of columns of F^{-1} .

Actual implementation

4.1 Starlet Transform

First how the starlet transform works, and then we discuss how we can leverage the properties to only use



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

*Figure 3: Scaling and wavelet function of the starlet wavelet. Source: [**?**]*

The starlet transform is an Isotropic Undecimated Wavelet Transform (IUWT) based on the starlet wavelet shown in figure 3. The IUWT decomposes an image into multiple layers, each representing parts of the image at different scales. The lower layers contain small structures like point sources, while the higher layers contain extended emissions like hydrogen clouds. The two transforms are shown in equation (4.2) and (4.3). The transformation from starlets to image x, shown in (4.2), is a simple addition of all J layers. The transformation from image x to starlet space, shown in (4.3) is done with a multi-scale wavelet decomposition.

$$\boldsymbol{x} = \boldsymbol{w_0} + \boldsymbol{w_1} + \ldots + \boldsymbol{w_{J-1}} + \boldsymbol{c_J} \tag{4.2}$$

$$\begin{array}{lll} c_0 = \boldsymbol{x} \star S_0 & c_1 = c_0 \star S_1 & \\ \boldsymbol{w_0} = \boldsymbol{x} - c_0 & \boldsymbol{w_1} = c_0 - c_1 & \end{array} \quad \ldots \quad \begin{array}{ll} c_{J-1} = c_{J-2} \star S_{J-1} & \\ \boldsymbol{w_{J-1}} = c_{J-2} - c_{J-1} & \boldsymbol{c_J} = c_{J-1} \star S_J \end{array} \tag{4.3}$$

Let us look at the multi-scale wavelet decomposition in more detail. w_i represents the starlet components at layer i, where w_0 contains the smallest structures in the image, and w_{J-1} contains the largest structures. Note that the starlets w_i, c_i and the image x all have the same dimensions. c_i contains an ever increasingly blurred version of the image x. c_i represents the image convolved with all the scaling functions $S_0, S_1, \ldots, S_i$, where S_i is the scaling function shown in figure 3a of the layer i. The scaling function increases in size for each successive layer.

The decomposition starts iteratively, blurrs the image x with the smallest scaling function S_0.

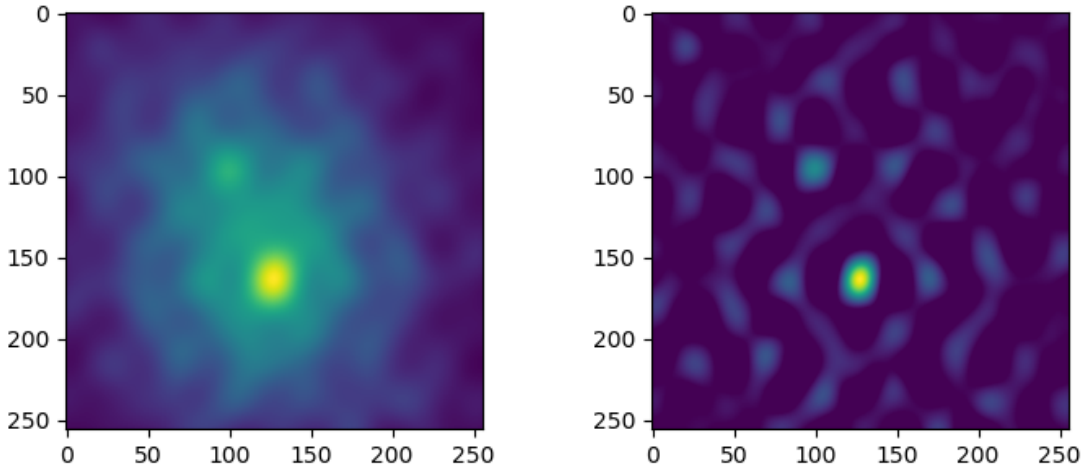
The last, heavily blurred c_J contains the structures which were too big to represent with the largest starlets, and is the last layer of the decomposition.

This is an over-complete representation. Many different components can explain the same pixel.

!!!!!A trous algorithm, size of the differtent S_i

4.1.1 Active set heuristic with Starlets

The multi-scale decomposition can be used to estimate the which starlet components w_i are likely to be non-zero. The figure 4 shows an example of the process. Each ”pixel” in 4b represents a starlet at scale 0, i.e. point sources. We can apply the shrink operator on the w_0 map, is essentially a parameter estimation. This property is used in the SASIR[**?**] reconstruction, which uses the Fast Iterative Shrinkage-Thresholding Algorithm (FISTA) for optimization.



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

Figure 4: Dirty map and w_0 starlet layer. The right image shows the probability distribution for point sources in the dirty image.

For our purposes, we use the result of the decomposition as a probability distribution. The higher the values of the decomposed w_i, the more likely it is to be non-zero.

the X highest values of the w_i decomposition are put into the active set. Only components in the active set get optimized.

Note however that the different locations are dependent on each other. The image 4b shows two areas for the two point sources. From the w_0 map alone it is hard to know if the center blob is due to a single point source,

or multiple point sources in a small area.

4.2 Coordinate Descent Implementation

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 (4.4) 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.

$$\underset{\alpha}{minimize} \left\| V - F^{-1}D\alpha \right\|_2^2 + \lambda \left\| \alpha \right\|_1 \tag{4.4}$$

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.

4.2.1 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

4.2.2 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 Coordinate Descent Reconstructions on Simulated Data

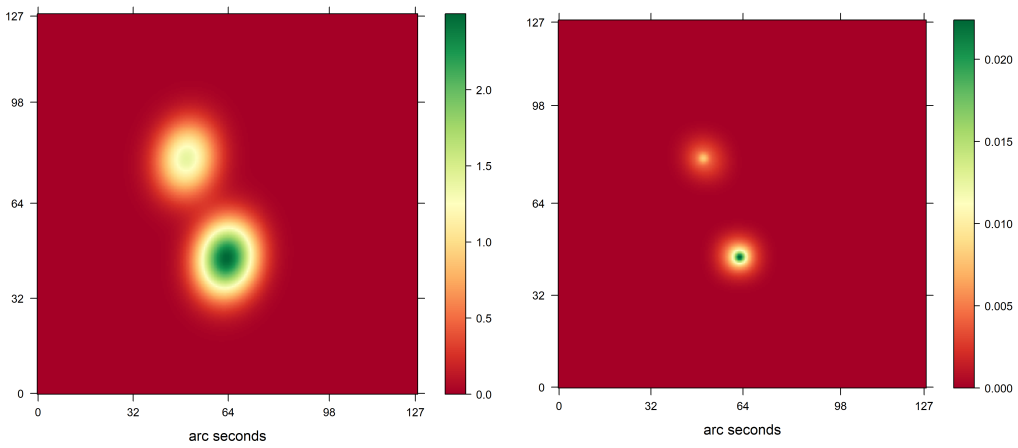
Here we test the Coordinate Descent algorithm on two simulated datasets and show we indeed only need a subset of the Fourier Transform Matrix' columns F^{-1}.

WE compare the reconstruction quality to CASA's CLEAN implementation.

Show super resolution, and more accurate flux reconstruction on simulated data.

We compare the reconstruction of Coordinate Descent and CLEAN on two simulated observations.

5.1 Super-resolution of two point sources



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

Figure 5: Image reconstruction of two point sources.

192 non-zeros

5.2 Flux reconstruction of extended Gaussian sources

v

In this section, we compare the Coordinate Descent reconstruction with CLEAN on simulated MeerKAT data, and compare the runtime complexity of the two approaches on a real-world MeerKAT observation. We show that Coordinate Descent reconstructs the image by only computing a subset of the Fourier Transform Matrix' columns F^{-1}, and investigate if this approach reduces the runtime complexity on the large scale reconstruction problems of MeerKAT.

The real world MeerKAT data were calibrated and averaged down to reduce its size to 88 Gigabytes. The raw, uncalibrated data ranges between 500 and 1000 Gigabytes. Data on this scale requires a mature pipeline for reading and image reconstruction. Within the time limit of this project, only a reconstruction with the WSCLEAN[**?**] pipeline was possible.

The simulations were created with the Meqtrees software package. Two simulations which contain roughly (size of Visibilities) perfectly calibrated Visibilities were created. Compared to real-world observations, the two simulated data sets are small and not representative of the real data volume. Also, more realistic simulations

which contain pointing errors, calibration errors, 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.

5.3 Imaging on Simulated Data

We compare the reconstruction of Coordinate Descent and CLEAN on two simulated observations. The first observation contains two point sources, on which we show that Coordinate Descent is able to locate the sources below the accuracy limit of the instrument. The second observation contains point and Gaussian sources, on which we show that Coordinate Descent better captures the intensity profile of extended emissions. Also, we look at the shortcomings of the proof-of-concept implementations in terms of reconstruction quality and runtime. We use the CLEAN implementation of CASA in this section. CASA is an established software framework for radio interferometer image reconstruction and was already used in a previous project.

We used CASA's default parameters for CLEAN, except for the maximum number of CLEAN iterations, which we set to 250. The proof-of-concept Coordinate Descent implementation has three parameters to tune: Number of iterations, the number of starlet layers J, and the regularization parameter λ. The first two parameter could be estimated by the reconstruction algorithm itself. In this implementation, it was left to the user. For the two simulations the Coordinate Descent parameters were chosen:

- Two Point sources: 4 full iterations, $J = 3$, $\lambda = 0.1$
- Mixed sources: 4 full iterations, $J = 7$, $\lambda = 0.1$

In figure 5 shows an image of the Coordinate Descent and CLEAN reconstruction. Figure 6 shows the intensity profile of both reconstructions compared with the ground truth. Note that figure 6 has a logarithmic y axis. The reconstructions differ in two notable ways: The peak intensity of the source, and how much each algorithm spreads the point source.

Spread: CLEAN reconstructs a convolved version of the true image. CLEAN locates the two peaks in figure 6, but convolves the point source with a Gaussian function. CLEAN reconstructs two Gaussian functions with same peak as the point source. The Gaussian represents the accuracy of the instrument. With compressed sensing, we try to reconstruct the observed image above the accuracy of the instrument. In the intensity profile, Coordinate Descent reconstructs two narrow gaussian-like peaks, essentially super-resolving the image. However, note that the total intensity is a fraction of the original peaks.

Intensity: [Total Flux] The reconstructions differ in intensity. The figure 6 shows the intensity pr. Also note that Coordinate Descent has shifted the smaller peak by a pixel in its reconstruction. [Total flux correct in CD?]

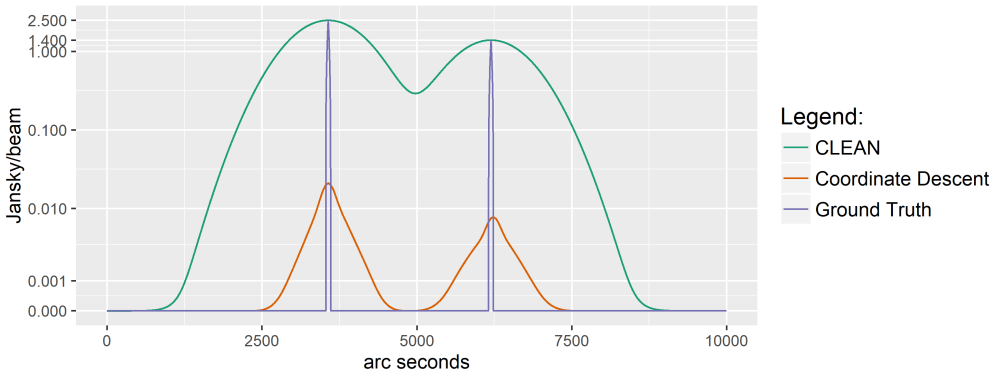
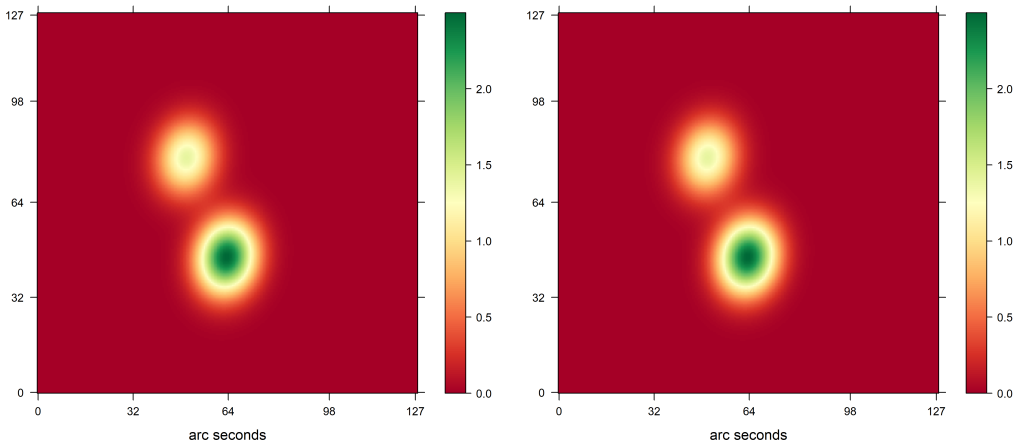


Figure 6: Intensity profile of the two point sources.

The difference in intensity becomes more apparent with extended emissions. Figure 7 shows Coordinate Descent and tCLEAN on the simulation with mixed sources. Again the two reconstructions arrive at different intensities. The gaussian emissions are reconstructed with a higher intensity than Coordinate Descent.



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

Figure 7: Reconstruction on mixed sources

With starlets, coordinate descent has a representation for extended emissions. Looking at the intensity profile of an extended emisions **??**, we see Coordiante Descent coming closer to the true intensity. Although in the four iterations, [it still had a considerable margin on error].

Total flux constraint, λ for different starlet layers like in [**?**]

Coordinate Descent did not reconstruct all point sources. How many starlets are non-zero is the major point for runtime. It depends on how many areas of the image are non-zero. Starlet has a representation for extended emission, how many starlets are needed for modelling is hard.

Runtime problems 1900 non-zero starlets,

6 Runtime complexity on large scale reconstructions

Pinning down the average runtime of Coordinate Descent is difficult. As mentioned in section 4, Coordinate Descent has lose convergence guarantees in theory, but works well in practice with heuristics. For our algorithm, the runtime mainly depends on the number of non-zero starlet components S and how many iterations Coordinate Descent needs to converge. How many non-zero starlets, or how many iterations Coordinate Descent needs on average are hard questions and cannot be answered in this project.

Instead, this work focuses on a minimum runtime estimate of Coordinate Descent on MeerKAT observations. We compare the runtime complexity with CLEAN and analyse what speed up an idealized version of Coordinate Descent can provide. Sadly, the speed up of an idealized Coordinate Descent is negligible. The Major Cycle architecture leads to a lower runtime complexity in the context of MeerKAT reconstructions.

6.1 Runtime complexity of an idealized Coordinate Descent

The runtime complexity of Coordinate Descent depends largely 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 best case scenario, we assume an oracle performance heuristic: It finds the location and number of the S non-zero starlet components in constant time ($O(1)$). Coordinate Descent therefore has to find the value of the S non-zero starlet components, which takes three separate operations: creating the columns of F^{-1}, calculating the minima for each single component, and calculating the starlet layers:

$$\begin{aligned} \text{creating } S \text{ columns of } F^{-1} &: S * 7M \\ \text{locating } S \text{ minima of } S \text{ parabolas} &: S * 4M \\ \text{calculating } J \text{ Starlet layers} &: J * 2M \end{aligned}$$

We assume we have enough memory to cache the columns of F^{-1} and only need to calculate them once. Coordinate Descent arrives at the correct result in I_{CD} iterations. Therefore we arrive at the runtime complexity of (6.1).

$$CD(I_{CD}, M, S, J) = S * 7M + I_{CD} * [S * 4M + J * 2M] \tag{6.1}$$

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 (pixels per arc-second). The runtime only depends indirectly on the image resolution, not the total number of pixels.

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 and a large amount of distinct structures, which leads to a large S (each point source needs at least one non-zero component to be represented).

6.2 Runtime complexity of CLEAN

We look at CLEAN reconstructions which use the non-uniform FFT with w-stacking. The runtime of a single Major cycle depends on the non-uniform FFT with w-stacking and the number of CLEAN deconvolutions. The

number N denotes the number of pixels (for example 512^2).

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

The overall complexity shown in (6.2) can also be split into two parts: It depends on the number of Major on the number of Major Cycles I_{Major} and the complexity of the non-uniform FFT, and I_{Major} times the CLEAN deconvolutions.

$$\begin{aligned} CLEAN(I_{Major}, I_{CLEAN}, M, N, W) = I_{Major} * 2 * [M + W * (2Nlog2N + 2N) + NlogN] \\ + I_{Major} * [I_{CLEAN} * 2N] \end{aligned} \tag{6.2}$$

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 N instead of the number of Visibilities M. 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 overall runtime.

6.3 Comparison on a MeerKAT reconstruction problem

The MeerKAT observation contains approximately 540 channels with 4 million calibrated Visibilities each. After calibration, MeerKAT data is typically averaged over frequency and time to reduce its disk space. There are no strict rules on how to chose the image resolution and number of pixels. An initial WSCLEAN reconstruction was preformed with:

- Visibilities: $M = 2.19e^9$
- w-stacks: $W = 128$
- Pixels: $N = 8192^2$
- Maximum number of CLEAN iterations: $I_{CLEAN} = 35'000$

For our estimate, we use the values of the initial WSCLEAN reconstruction, and we assume CLEAN requires $I_{Major} = 10$ Major Cycles to converge. CLEAN reconstructions tend to use around five Major Cycles. Compressed sensing reconstructions which use the Major Cycle framework tend to use around 10 cycles. For the first estimate, we make assumptions in favour of Coordinate Descent.

The idealized Coordinate Descent algorithm has three parameters left: Number of starlet levels J, number of non-zero starlets S, and number of iterations to converge I_{CD}. We under-estimate and set $J = 5$, and assume the largest structure in the image is no more than 160 pixels in size. With these assumptions we plug in the values into the equations (6.1) and (6.2), and compare the runtime for different values of S and I_{CD} in table 1

To put the numbers into perspective, Coordinate Descent used about 2000 non-zero starlets to reconstruct the image 7b. Since the reconstruction does not contain all point sources, the true number of S is likely to be higher than 2000 on a simulated dataset with simple, Gaussian extended emissions. A MeerKAT reconstruction can contain more complex emissions. Figure 8 shows the WSCLEAN reconstruction of the current observation. We see a large number of small, point-like objects and several complex extended emissions. We

	$I_{CD} = 1$	$I_{CD} = 5$	$I_{CD} = 10$	$I_{CD} = 15$
$S = 1000$	2.35	1.12	0.66	
$S = 2000$	1.17	0.56	0.33	
$S = 3000$	0.78	0.37	0.22	
$S = 4000$	0.59	0.28	0.17	

Table 1: Relative speed-up of an idealized Coordinate Descent compared to CLEAN.

need at least 2000 non zero starlet components to represent the MeerKAT image. In the best case scenario, when the ideal Coordinate Descent converges within one iteration, we have a 17% runtime improvement over CLEAN. Keep in mind that we have over-estimated the runtime of CLEAN.

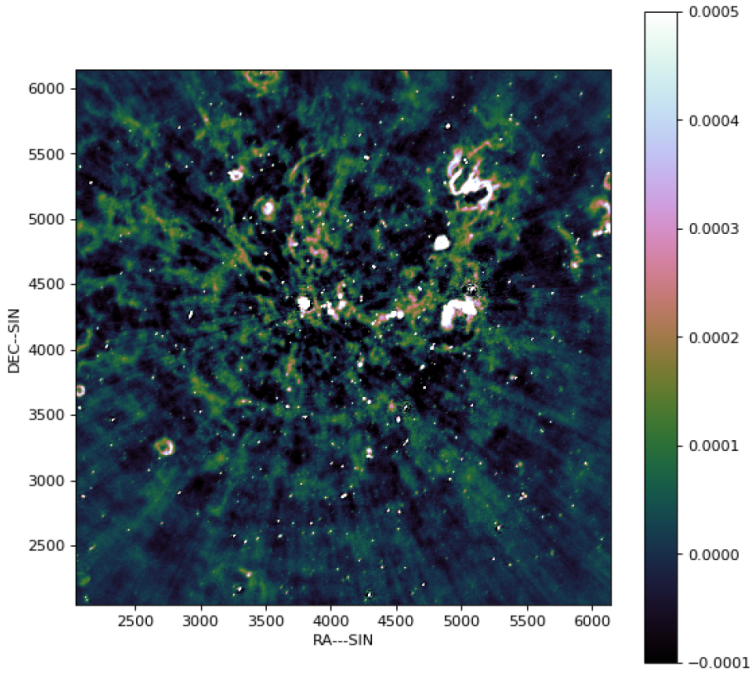


Figure 8: WSCLEAN Reconstruction of the MeerKAT observation.

Memory requirement,.

Let us ignore the memory requirement. Coordinate Descent does not scale with the number of pîxels. since this number is typically left to the user, the question remains if Coordinate Descent is worth the costs if we have a lot more pixels. We do a similar estimate, but quadruple each image dimension to $N = 32768^2$ and compare the speed up to CLEAN in table 2.

Half as many pixels as Visibilities. An extreme example.

when we say we need at least $S = 2000$, and at least as many iterations for reconstruction. Since Coordinate Descent does not depend on the image size N, and the image size changes depending on the use case, one might imagine Coordinate Descent may be useful for very large image sizes. Indeed, it is

but sadly, even the Sadly, this is not the case. Let us quadruple each image dimension to $N = 32000^2$, now we have two Visibilities for each pixel, and just for demonstration's sake reduce $J = 5$ for Coordinate Descent. The results are shown in table 2. Coordinate Descent is only competitive if it needs just one iteration to converge. Any advantage shrinks fast as soon as Coordinate Descent needs several iterations over all non-zero components S. How likely is it that Coordinate Descent just needs one iteration to converge?

	$I_{CD} = 1$	$I_{CD} = 5$	$I_{CD} = 10$	$I_{CD} = 15$
$S = 1000$	38.45	18.37	10.83	
$S = 2000$	19.23	9.19	5.42	
$S = 3000$	12.82	6.13	3.62	
$S = 4000$	9.62	4.60	2.71	
$S = 10000$	3.85	1.84	1.09	
$S = 15000$	2.57	1.23	0.72	
$S = 20000$	1.92	0.92	0.54	

Table 2: Relative speed-up of Coordinate Descent compared to CLEAN with an image size of $N = 32000^2$.

The Coordinate Descent approach sees considerable speed up, if it converges within one iteration. But it falls off quickly as soon as it needs several iterations to converge. For reference, the proof-of-concept approach uses at least ten iterations.

The problem is MeerKAT can always just put in more Visibilities. M can be increased at will. As soon as we go into self-calibration, we are hopeless. The number of Visibilities increases by factor 4-16, and Coordinate Descent is again too expensive to run.

The issue with Coordinate Descent's runtime complexity lies in the term $I_{CD} * [S * 4M + \ldots]$ of (6.1). 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.

We cannot move far away from the ideal algorithm before the speed up vanishes.

We over-estimated the runtime of CLEAN with 10 Major Cycles. If we assume 5 Major Cycles, which is what we would more realistically expect from clean, we essentially half the speed up in tables 1 and 2. Our CLEAN estimate puts the runtime complexity in the ballpark of other compressed sensing reconstructions. In the MeerKAT context where we have a factor of 10-20 times more Visibilities than pixels, the Major cycle improves the runtime complexity of reconstruciton algorithms

Indeed, the runtime of our Coordinate Descent algorithm can be improved by using the major cycle architecture, essentially replacing M with N and we arrive at the term h $I_{CD} * [S * 4N + \ldots]$. 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^{-1}, and reduce the memory requirement to a sane amount.

6.4 Embracing the Major Cycle

We searched for a compressed sensing algorithm which does not need the major cycle, with the hope it would improve runtime complexity. Arriving at a Coordinate Descent algorithm, which does not require any of the approximation algorithm. It is very simple to implement. Results of

Runtime estimate

Ironically, we can also use the major cycle in this algorithm and improve the runtime, maybe even get rid of the large memory requirement.

Can we use Coordinate Descent for asynchronous optimization.

Huge memory requirement. In the context of MeerKAT, it does not improve runtime compared to other compressed sensing approaches, let alone clean.

Self-calibration increases the

At least for MeerKAT, the signs point to the Major Cycle architecture to handle large scale image reconstruction problems. From different directions, self-calibration, we always arrive at a similar Major Cycle Architecture for image reconstruction.

In this project explored different architectures for compressed sensing reconstructions.

We did this and arrived at a simple proof-of concept Coordinate Descent algorithm, getting rid of the major cycle. Although it again may produce better reconstructions,

The major cycle seems here to stay. As soon as we factor in self-calibration, the reconstruction algorithm has to transform between Visibilities and image space several times. Becoming some form of major cycle.

We have developed an algorithm which does not need the major cycle. But because it does not have one, it scales worse on large scale reconstruction problems of MeerKAT.

The question still remains open., how we get compressed sensing to scale better and finally getting rid of clean.

Moving to the sphere.

Asynchronous implementations.

7 Compressed Sensing reconstructions in the MeerKAT era

Runtime is a major drawback of Compressed Sensing reconstructions.

This project searched if the architecture can be optimized for Compressed sensing.

How to get the runtime down at least to comparable levels of CLEAN is still an open question. Still an open question

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 18, 2019

Jonas Schwammberger