PDR.02.05.03 Imaging Pipeline
<table>
<thead>
<tr>
<th>Name</th>
<th>Designation</th>
<th>Affiliation</th>
</tr>
</thead>
<tbody>
<tr>
<td>P. Alexander</td>
<td>SDP lead</td>
<td>University of Cambridge</td>
</tr>
<tr>
<td>R. Nijboer</td>
<td>SDP.PIP lead</td>
<td>ASTRON</td>
</tr>
</tbody>
</table>

**Signature & Date:**

**Signature:**

<table>
<thead>
<tr>
<th>Name</th>
<th>Signature:</th>
<th>Email:</th>
</tr>
</thead>
<tbody>
<tr>
<td>P. Alexander</td>
<td>Paul Alexander (Feb 9, 2015)</td>
<td><a href="mailto:pa@mrao.cam.ac.uk">pa@mrao.cam.ac.uk</a></td>
</tr>
<tr>
<td>R. Nijboer</td>
<td>Ronald Nijboer (Feb 9, 2015)</td>
<td><a href="mailto:rnijboer@astron.nl">rnijboer@astron.nl</a></td>
</tr>
</tbody>
</table>

**Version** | **Date of issue** | **Prepared by** | **Comments**

| 1.0         | 2015-02-09     | A. Scaife       | Unrestricted |

**ORGANISATION DETAILS**

| Name | Science Data Processor Consortium |
Table of Contents

List of abbreviations 5
List of symbols 6
List of figures 7
List of tables 8
Summary 9

Applicable and reference documents 10

1 Introduction 12

2 Pipeline Overviews 13
  2.1 Continuum Imaging Pipeline ............................................. 13
     2.1.1 Description ......................................................... 13
  2.2 Spectral Imaging Pipeline ................................................. 15
     2.2.1 Description ......................................................... 15
  2.3 Slow Transients Pipeline ................................................ 18
     2.3.1 Description ......................................................... 18

3 Common Elements 23
  3.1 Gridding ................................................................. 23
     3.1.1 Description ......................................................... 23
     3.1.2 Cost ................................................................. 24
     3.1.3 Scalability ........................................................ 24
  3.2 W-projection .............................................................. 25
     3.2.1 Description ......................................................... 25
     3.2.2 Cost ................................................................. 27
     3.2.3 Scalability ........................................................ 30
  3.3 A-projection .............................................................. 30
     3.3.1 Description ......................................................... 30
     3.3.2 Cost ................................................................. 32
     3.3.3 Scalability ........................................................ 34
  3.4 Fast Fourier Transform .................................................. 34
     3.4.1 Description ......................................................... 34
     3.4.2 Cost ................................................................. 34
     3.4.3 Scalability ......................................................... 35
  3.5 Sparse Fourier Transform ................................................ 36
     3.5.1 Description ......................................................... 36
     3.5.2 Cost ................................................................. 37
     3.5.3 Scalability ......................................................... 38
  3.6 Major Cycle .............................................................. 38
     3.6.1 Description ......................................................... 38
     3.6.2 Cost ................................................................. 41
     3.6.3 Memory Requirements ............................................. 43
     3.6.4 Scalability ......................................................... 43
List of abbreviations

GSM  Global Sky Model
LSM  Local Sky Model
FFT  Fast Fourier Transform
SFT  Sparse Fourier Transform
### List of symbols

- $N_{\text{major}}$: Number of Major Cycles
- $N_{\text{minor}}$: Number of Minor Cycles
- $N_{\text{vis}}$: Number of continuous visibilities
- $N_{\text{pix}}$: Number of pixels on side of image
- $N_c$: Number of frequency channels
- $N_b$: Number of beams
- $N_{\text{kernel}}$: Support in pixels of gridding convolution kernel
- $N_A$: Support in pixels of Mueller matrix element
- $N_{\text{pol}}$: Number of polarisations
- $N_{\text{cvff}}$: Support of over-sampled convolution kernel
- $N_{\text{freq}}$: Number of frequencies
- $N_{\text{basel}}$: Number of baselines
- $N_{\text{facet}}$: Number of facets
- $N_{\text{stokes}}$: Number of Stokes parameters
- $N_S$: Number of scales
- $N_t$: Number of Taylor terms
- $N_{\text{mm}}$: Factor to account for inclusion of off-diagonal Mueller matrix terms
- $N_{\text{inc}}$: Number of parallactic increments
- $N_{\text{dec}}$: Decimation factor
- $N_{\text{time}}$: Number of time steps
- $N_{K,S}$: Support of convolution kernel for multi-scale imaging
- $\mathbf{I}^M$: Model image
- $\mathbf{A}^M$: Model primary beam
- $\mathbf{A}^{\text{Obs}}$: Measured primary beam
- $\mathbf{V}^M$: Model visibilities
- $\mathbf{V}^{\text{Obs}}$: Measured visibilities
- $\mathbf{V}^{\text{Res}}$: Residual visibilities, such that $\mathbf{V}^{\text{Res}} = \mathbf{V}^{\text{Obs}} - \mathbf{V}^M$
- $\tau_{\text{ss}}$: Length of snapshot
- $\tau_{\text{obs}}$: Length of observation
- $\theta_{\text{FOV}}$: Field of View in angular units
- $\theta_{\text{res}}$: Resolution element in angular units
- $Q_{\text{GCF}}$: Over-sampling for convolution kernel
## List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Continuum Imaging Pipeline Data Distribution Overview</td>
</tr>
<tr>
<td>2</td>
<td>Spectral Imaging Pipeline Data Distribution Overview</td>
</tr>
<tr>
<td>3</td>
<td>Slow Transients Pipeline Data Distribution Overview</td>
</tr>
<tr>
<td>4</td>
<td>Division of Operations for Slow Transients Pipeline</td>
</tr>
<tr>
<td>5</td>
<td>SKA1 $w$-kernel support sizes as a function of $w$.</td>
</tr>
<tr>
<td>6</td>
<td>$w$-kernel function for an SKA1-SURVEY-like array</td>
</tr>
<tr>
<td>7</td>
<td>ASKAPsoft $w$-projection kernel size.</td>
</tr>
<tr>
<td>8</td>
<td>Comparison of MPI-enabled Imaging</td>
</tr>
<tr>
<td>9</td>
<td>Dynamic range as a function $A$-kernel update rate.</td>
</tr>
<tr>
<td>10</td>
<td>Performance comparison of Fourier Transform Algorithms.</td>
</tr>
<tr>
<td>11</td>
<td>MS-MFS Flow Diagram</td>
</tr>
<tr>
<td>12</td>
<td>CS Flow Diagram</td>
</tr>
<tr>
<td>13</td>
<td>MS-MFS Data Distribution Diagram - CASA-style.</td>
</tr>
<tr>
<td>14</td>
<td>MS-MFS Data Distribution Diagram - ASKAP-style.</td>
</tr>
<tr>
<td>15</td>
<td>Proposed MS-MFS Interconnects.</td>
</tr>
<tr>
<td>16</td>
<td>CS Data Distribution Diagram.</td>
</tr>
<tr>
<td>17</td>
<td>Parameter space for the detection of pulsed transient radio sources: pulse width against dispersion measure.</td>
</tr>
<tr>
<td>18</td>
<td>Data Product Creation Data Distribution Diagram.</td>
</tr>
</tbody>
</table>
## List of Tables

1. Summary comparison of Fourier Transform algorithms ........................................ 21
2. Dynamic range vs. computation saved when removing highest \( w \)-points. ....... 28
3. \( A \)-kernel parameters for various arrays ................................................... 33
4. Summary of De-dispersion characteristics. ...................................................... 53
Summary

This document provides additional detail on the design of the three SKA1 imaging pipelines, the common elements used to construct those pipelines and supporting information to justify design recommendations.
Applicable and reference documents

Applicable Documents

The following documents are applicable to the extent stated herein. In the event of conflict between the contents of the applicable documents and this document, the applicable documents shall take precedence.

<table>
<thead>
<tr>
<th>Reference no.</th>
<th>Reference</th>
</tr>
</thead>
</table>

Reference Documents

The following documents are referenced in this document. In the event of conflict between the contents of the referenced documents and this document, this document shall take precedence.

<table>
<thead>
<tr>
<th>No.</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>[RD05]</td>
<td>N. Mathur (1969), A pseudodynamic programming technique for the design of correlator supersynthesis arrays, Radio Science, vol. 4</td>
</tr>
<tr>
<td>No.</td>
<td>Reference</td>
</tr>
<tr>
<td>-----</td>
<td>-----------</td>
</tr>
<tr>
<td>[RD27]</td>
<td>Tim Cornwell, Ben Humphreys, Emil Lenc, Maxim Voronkov, Matthew Whiting (2011), ASKAP Science Processing, ASKAP-SW-0020</td>
</tr>
</tbody>
</table>
1 Introduction

This document provides additional detail on the design of the three SKA1 imaging pipelines, the common elements used to construct those pipelines and supporting information to justify design recommendations.
2 Pipeline Overviews

2.1 Continuum Imaging Pipeline

2.1.1 Description

The purpose of the Continuum Imaging Pipeline is to transform the calibrated continuous visibility data into an “image”. For the Continuum Imaging Pipeline, this image data product will have the dimensions $N_{\text{pix}} \times N_{\text{pix}} \times N_t$; it will contain the first $N_t$ Taylor terms of a frequency dependent expansion of the data in image space. Additional products include a “residual image”, “model image” and a representative PSF at a nominal frequency.

The Continuum Imaging Pipeline is based around a Major Cycle / Minor Cycle deconvolution process. Following terminology established by [RD01], the Minor Cycle represents the update to the model image. The Minor Cycle typically operates in the Stokes frame. Here we re-cast this definition to establish the Minor Cycle as that part of the Major Cycle which updates the model in the Stokes frame. This does not affect existing descriptions of CLEAN based deconvolution methods, but allows us to describe Compressed Sensing (CS) deconvolution methods in the Major/Minor Cycle framework.

The steps of the Continuum Imaging Pipeline are:

1. Initialisation of intermediate image and visibility products;
2. Initialisation of convolution kernels (for use within the Major Cycle);
3. $N_{\text{major}}$ Major Cycles, each containing $N_{\text{minor}}$ Minor Cycles;
4. Output of final data products.

This sequence of steps assumes that convolution kernels will be calculated once and cached. If insufficient memory is available for this purpose, these kernels must be re-calculated during each Major Cycle.

A generic overview of data distribution within this pipeline is shown in Fig. 1. The exact details of a specific data distribution will depend on the form of the Major Cycle. Fig. 1 illustrates the case of an image-plane deconvolution such as the basic Högbom CLEAN [RD02]. More detailed distribution diagrams are presented for specific deconvolution methods in § 3.6.4.

During the Major Cycle, the dominant processing steps are (1) the formation of a residual image in order to enable the use of the Minor Cycle and update the model image. This is known as the ‘backward’ step as the visibility data is being propagated back to the image domain:

$$I^R = [\Phi^\dagger \Phi] \Phi^\dagger V^R,$$

where $\Phi$ is the measurement operator; (2) the calculation of model visibilities from the model image, known as the ‘forward’ step:

$$V^M = \Phi I^M.$$

The forward calculation must be done with high accuracy, but since the overall approach to deconvolution is iterative, the backward calculation may be performed with lower accuracy [RD03;RD04].

The measurement operator, $\Phi$, includes a convolutional gridding component and a Fourier transform component. The convolution kernel used in gridding has multiple components.
Figure 1: Continuum Imaging Pipeline Data Distribution Overview.
The components of the Continuum gridding kernel include:

1. an anti-aliasing term;  
   Dependencies: frequency;
2. $w$-term for correcting effects of non-co-planarity;  
   Dependencies: time, frequency, baseline;
3. $A$-term for correcting direction-dependent effects;  
   Dependencies: time, frequency, baseline.

Further details of these kernels are contained in the later sections of this document.

**Recommendations** For the Continuum Imaging Pipeline we recommend that

- Initial compute island data distribution is done in frequency, with time distribution being done intra-island.
- Both $w$- and $A$- projection are implemented in the gridding/de-gridding steps. The justification for this choice is the dynamic range requirements of the telescope.
- Imaging incorporates the $w$-snapshots method to reduce the support of the $w$-kernel. The justification for this choice is the computational load of the gridding step which scales as $N_{\text{kernel}}^2$ and dominates the computation for the Continuum Imaging Pipeline.
- A degree of faceting is incorporated into the Major Cycle to reduce the support of the $w$-kernel. The justification for this choice is (1) the computational load of the gridding step which scales as $N_{\text{kernel}}^2$ and dominates the computation for the Spectral Imaging Pipeline; (2) the potential for distributing the facets across compute nodes for much of the Major Cycle.
- The $A$-projection incorporates the off-diagonal, but not the anti-diagonal, elements of the Mueller matrix. The justification for this choice is the dynamic range requirements of the telescope. This recommendation requires further empirical verification.
- Deconvolution is performed using the MS-MFS method. The justification for this choice is the current lack of frequency dependent CS reconstruction methods.

**2.2 Spectral Imaging Pipeline**

**2.2.1 Description**

The purpose of the Spectral Imaging Pipeline is to transform the calibrated continuous visibility data into an “image”. For the Spectral Imaging Pipeline, this image data product will have the dimensions $N_{\text{pix}} \times N_{\text{pix}} \times N_{\text{chan}}$; it will contain images at $N_{\text{chan}}$ individual frequencies. Additional products include a “residual image” cube, a “model image” cube and a representative PSF at a nominal frequency.

Unlike the Continuum Imaging Pipeline, the Spectral Imaging Pipeline is not expected to produce a frequency dependent model for the sky intensity but rather a multi-frequency image cube. The Spectral Imaging Pipeline will be run on a dataset following the Continuum Imaging
Pipeline. The frequency dependent model produced by the Continuum Imaging Pipeline will be used to update the Global Sky Model (GSM) for the field and form a Local Sky Model (LSM). This LSM will then be subtracted from the continuous visibilities over the selected spectral window identified for analysis.

<table>
<thead>
<tr>
<th>The steps of the Spectral Imaging Pipeline are:</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Initialisation of intermediate image and visibility products;</td>
</tr>
<tr>
<td>2. Extract spectral window of interest from full visibility dataset;</td>
</tr>
<tr>
<td>3. Subtract LSM from reduced visibility dataset;</td>
</tr>
<tr>
<td>4. (Optional) Identify sub-regions (“postage stamps”) of interest;</td>
</tr>
<tr>
<td>5. Initialisation of convolution kernels (for use within the Major Cycle);</td>
</tr>
<tr>
<td>6. $N_{\text{major}}$ Major Cycles, each containing $N_{\text{minor}}$ Minor Cycles;</td>
</tr>
<tr>
<td>7. Output of final data products.</td>
</tr>
</tbody>
</table>

This sequence of steps assumes that convolution kernels will be calculated once and cached. If insufficient memory is available for this purpose, these kernels must be re-calculated during each Major Cycle.

A generic overview of data distribution within this pipeline is shown in Fig. 2. The exact details of a specific data distribution will depend on the form of the Major Cycle. The components of the Major Cycle are identical to those of the Continuum Imaging Pipeline with $N_i = 1$.

For wide-field spectral imaging both non-coplanarity and direction-dependent effects may need to be accounted for in the gridding component of the measurement operator. However, it may be possible to neglect the $\lambda$-kernel in gridding as dynamic range limitations are not as severe as for the continuum case.

<table>
<thead>
<tr>
<th>The components of the wide-field Spectral gridding kernel include:</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. an anti-aliasing term;</td>
</tr>
<tr>
<td>Dependencies: frequency;</td>
</tr>
<tr>
<td>2. $w$-term for correcting effects of non-co-planarity;</td>
</tr>
<tr>
<td>Dependencies: time, frequency, baseline;</td>
</tr>
<tr>
<td>3. $\lambda$-term for correcting direction-dependent effects;</td>
</tr>
<tr>
<td>Dependencies: time, frequency, baseline.</td>
</tr>
</tbody>
</table>

For postage-stamp (small-field) spectral imaging it may be possible to neglect both the effects of non-coplanarity and direction-dependent gains.
Figure 2: Spectral Imaging Pipeline Data Distribution Overview.
The components of the postage-stamp Spectral gridding kernel include:

1. an anti-aliasing term;
   Dependencies: frequency.

**Recommendations**  For the Spectral Imaging Pipeline we recommend that

- Initial compute island data distribution is done in frequency, with time distribution being done intra-island.
- Both $w$- and $A$- projection are implemented in the gridding/de-gridding steps. The justification for this choice is the dynamic range requirements of the telescope.
- Imaging incorporates the $w$-snaps shots method to reduce the support of the $w$-kernel. The justification for this choice is the computational load of the gridding step which scales as $N_{\text{kernel}}^2$ and dominates the computation for the Continuum Imaging Pipeline.
- A degree of faceting is incorporated into the Major Cycle to reduce the support of the $w$-kernel. The justification for this choice is (1) the computational load of the gridding step which scales as $N_{\text{kernel}}^2$ and dominates the computation for the Spectral Imaging Pipeline; (2) the potential for distributing the facets across compute nodes for much of the Major Cycle.
- The $A$-projection incorporates the off-diagonal, but not the anti-diagonal, elements of the Mueller matrix. The justification for this choice is the dynamic range requirements of the telescope. This recommendation requires further empirical verification.
- Deconvolution is performed using either the MS-MFS or the CS method. The justification for this choice is (1) no multi-frequency synthesis is required; (2) the CS method is expected to be computationally less expensive for small fields of view if postage stamp imaging is implemented.

### 2.3 Slow Transients Pipeline

#### 2.3.1 Description

The purpose of the Slow Transients Pipeline is to transform the calibrated continuous visibility data into a catalogue of transient sources. Transient sources are defined as *any source with variable emission with respect to the GSM.* Additional products may include postage-stamp images of detections. The Slow Transients Pipeline will operate on snap-shots of data. The length of these snap-shots, $\tau_{ss}$, is expected to be of order $\sim$ 1 second; however, with no formal requirement on cadence this pipeline has been designed to run as close to correlator dump-time as achievable. In order to improve detection rates the Slow Transients Pipeline should maximise the available FOV. In order to improve cross-matching with data at other wavelengths the Slow Transients Pipeline should maximise resolution.

Catalogue entries from the ST Pipeline are expected to include:
1. Equatorial position;
2. Time stamp;
3. Flux density;
4. Spectral Index;
5. (recommended) Dispersion measure.

All measured quantities are expected to be listed with their associated errors; for flux density values, the associated error will be the local rms noise. Errors on spectral index will be calculated assuming that calibration uncertainty is 100% correlated across the frequency band.

Unlike the Continuum and Spectral Imaging Pipelines, the Slow Transients Pipeline does not necessarily need to form images. There are two potential implementations of this pipeline:

Option 1 For each snap-shot: subtract the GSM, grid the subtracted visibilities, form a dirty image via FFT, perform source detection on each image.

Option 2 For each snap-shot: subtract the GSM, grid the subtracted visibilities, use a thresholded sparse Fourier transform in place of an FFT on the gridded visibilities and recover a list of significant components.

A generic overview of data distribution within this pipeline is shown in Fig. 3, illustrating both options. It is assumed that no deconvolution step is required for this pipeline as transient sources are expected to be unresolved and isolated (sparse). However: see de-dispersion note in Recommendations.

The computationally dominant steps within both of these processes are the gridding and Fourier transform steps. Since the gridding operation scales as $N_{\text{vis}}$ and $N_{\text{vis}}$ scales linearly with time, the computational load for a given period of observation time in the Slow Transients Pipeline rapidly becomes dominated by the FFT step, which scales with image/grid size, $N_{\text{pix}}^2$. This is in contrast to the Continuum and Spectral Imaging Pipelines, where $N_{\text{vis}}$ will be a factor of $\tau_{\text{obs}}/\tau_{\text{ss}}$ larger and the Fourier Transform will be performed $\tau_{\text{obs}}/\tau_{\text{ss}}$ fewer times. This is illustrated for the three instruments in Fig. 4 where it can be seen that snapshot durations for minimum operations are significantly longer than those suitable for transient detection.

Choice of Fourier Transform method is therefore key to the performance of the Slow Transients Pipeline. Characteristics of the two Fourier Transform methods from Options (1) & (2) are summarised in Table 1 and described in more detail later in this document.

**Note:** Gather Point A in Figure 3 indicates the point at which data distributed in frequency should be gathered. Although this gather step is optional, its implementation is recommended due to the increase in signal-to-noise it provides for each facet in the source detection step. However, due to the high cadence of this pipeline the estimated data rates for the interconnect required to perform this reduction are very high. Using $uv$-grid data or image data for the reduction is equivalent as, although the $uv$-grid data are complex and the image data real, the $uv$-grid data are Hermitian and only half need to be used. However, a further advantage of the
Figure 3: Slow Transients Pipeline Data Distribution Overview.
Figure 4: Division of Operations between GRIDDING (red) and FFT (blue) as a function of snapshot duration for the Slow Transients Pipeline. Numbers are representative for a 6hr total observing duration, a $N_{\text{pix}} = 2^{12}$, $N_{\text{kernel}} = 16$ and assuming no time averaging relative to the correlator dump time (indicated by the pale blue vertical line) for each instrument. Indicative correlator dump times are assumed to be 80, 300 & 600 ms for SKA1-MID, SK1-SURVEY & SKA1-LOW, respectively.

Table 1: Summary comparison of Fourier Transform algorithms for the Slow Transients Pipeline.

<table>
<thead>
<tr>
<th></th>
<th>FFT</th>
<th>SFT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Complexity</td>
<td>$N_{\text{pix}}^2 \log_2 N_{\text{pix}}$</td>
<td>$V_t^{-3.4} N_{\text{pix}}^{1.5} \log_2 N_{\text{pix}}$</td>
</tr>
<tr>
<td>Relative Run-time ($N_{\text{side}} = 2^{12}$)</td>
<td>1.0</td>
<td>0.1</td>
</tr>
<tr>
<td>Relative Run-time ($N_{\text{side}} = 2^{15}$)</td>
<td>1.0</td>
<td>0.01</td>
</tr>
<tr>
<td>Efficiency (CPU)</td>
<td>8 – 15%</td>
<td>8 – 12%</td>
</tr>
</tbody>
</table>

SFT is that it only uses a small percentage of the total $uv$-grid data and this decimation can be done on each node before the reduction step.

Recommendations  For the Slow Transients Pipeline, we recommend that

- Initial compute island data distribution is done in frequency, with additional distribution being done intra-island.

- Following initial tests on simulated SKA1-scale data, we recommend Option (2) for the Slow Transients Pipeline. The justifications for this choice are (1) the improved run-time for large images; (2) the further reduction in computational load produced by avoiding the source-finding step; (3) the potential reduction in inter-island data rates.

- Furthermore, we suggest that the $A$-term in the gridding kernel is not required for the Slow Transients Pipeline. The justifications for this choice are (1) time-dependent beam effects are minimised in snap-shot images; (2) the dynamic range of the transient sky is expected to be intrinsically low. This recommendation remains to be empirically verified.

- We suggest that the run-time for the FFT on large-images is prohibitively slow for this pipeline, which is intended to produce alerts in real-time. The FFT-based run-time could be improved by forming images at poorer resolution by selecting only the inner portion of the Fourier plane to be transformed. However, this would compromise both signal-to-noise and localisation of detected sources.

- We suggest that a de-dispersion step could be incorporated into this pipeline to improve signal-to-noise for potential transient sources.
Consequently, we summarise the steps of the Slow Transients Pipeline below:

<table>
<thead>
<tr>
<th>The steps of the Slow Transients Pipeline (detection) are:</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. (Optional) Extract spectral window of interest from full visibility dataset;</td>
</tr>
<tr>
<td>2. Initialisation of intermediate visibility products;</td>
</tr>
<tr>
<td>3. Subtract GSM from reduced visibility dataset (Note 1);</td>
</tr>
<tr>
<td>4. Initialisation of convolution kernels (for use in the gridding step);</td>
</tr>
<tr>
<td>5. Grid continuous visibilities to regular grid;</td>
</tr>
<tr>
<td>6. (Optional) De-disperse in time;</td>
</tr>
<tr>
<td>7. Threshold-ed Sparse Fourier Transform (SFT) on each dispersion measure grid;</td>
</tr>
<tr>
<td>8. Output of final catalogue entry.</td>
</tr>
</tbody>
</table>

For quality control, we suggest that it may be advantageous to extract additional data products in the form of postage-stamp images for detected transient sources. These are expected to be small enough not to require a $w$- or $A$-term in the gridding convolution kernel. Such images can be formed from the reduced visibility dataset (Note 1 above).

<table>
<thead>
<tr>
<th>The steps of the Slow Transients Pipeline (postage-stamp) are:</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Identify cached visibility dataset;</td>
</tr>
<tr>
<td>2. Initialisation of intermediate visibility and image products;</td>
</tr>
<tr>
<td>3. Initialisation of convolution kernels appropriate for postage-stamp;</td>
</tr>
<tr>
<td>4. Grid continuous visibilities to regular grid;</td>
</tr>
<tr>
<td>5. (Optional) De-disperse according to DM of detection;</td>
</tr>
<tr>
<td>6. FFT to image plane;</td>
</tr>
<tr>
<td>7. Output postage-stamp image.</td>
</tr>
</tbody>
</table>
3 Common Elements

3.1 Gridding

3.1.1 Description

In order to make use of the Fast Fourier Transform to produce images, the visibility data must be transformed to a regular grid. A basic interpolation between sampled $V(u,v)$ data onto a rectangular grid results in undesirable artifacts for the inverse Fourier transform, which can be reduced by more sophisticated interpolation approaches. Early approaches to gridding radio astronomy data simply added each data point $V(u,v)$ to an accumulator at its nearest grid point [RD05], or averaged those values that fall nearest to each grid point [RD06]. The technique commonly known as the gridding method was first developed by [RD07] and utilised a weighted sum based on distance between the sample point $(u,v)$ and the grid point, such as a Gaussian average. In [RD08] convolution with a sinc function was shown to provide optimal gridding. However since a sinc function has infinite extent convolution kernels $C$ with finite support are instead used.

Often a prolate spheroidal function is chosen in radio astronomy, as they are band limited and useful for recovering time limited functions from their Fourier transform [RD09:RD10]. Where wide fields of view are to be imaged, additional components are included in the convolution kernel beyond the traditional anti-aliasing term. These are known as the $w$- and $A$-kernels and are described in the following sections.

As the $(u,v)$ are not uniformly distributed, using a sum at each grid point for the convolutions of each $V(u,v,w)$ give densely sampled regions in the $uv$ plane more weight, which can be corrected by dividing the $V(u,v,w)$ by the convolution of $S(u,v)$ with the convolution function $C(u,v)$. Also, as the convolution function may attenuate the sides of the resulting image, the image can be divided by the inverse Fourier transform of the convolution function. If the grider is denoted by $III(u,v) = \sum \delta(u-x)\delta(v-y)$ then the image produced by such a gridding process can be written mathematically as:

$$I(l,m) = \frac{F^{-1}\left(\left(\frac{V(u,v)S(u,v)}{S(u,v)} \ast C(u,v)\right) \cdot III(u,v)\right)}{F^{-1}(C(u,v))}.$$ 

This is described in [RD11] which also compares different convolution functions: cosine, Gaussian, prolate spheroidal functions, and a simpler to compute Kaiser-Bessel function alternative.

A naïve approach to gridding would simply apply a chosen convolution kernel to each visibility in each of its channels. One basic efficiency that can be made is that the same convolution entries can be used for the visibilities in each polarisation pair XX, XY, YX, YY:

**GENERAL GRIDDING**

Initialise gridXX, gridXY, gridYX, gridYY accumulators to 0
for each visibility sample $V(u,v,w)$ do
  for each channel $f$ do
    $x = \text{round}(u), y = \text{round}(v)$ calculate nearest position on grid
    $C = \text{getConvolutionKernel}(u,v,w) \quad \text{in general kernel might not be fixed}$
    for each kernel column $i$ do
      $i$ goes from $-(\text{width}-1)/2$ to $(\text{width}-1)/2$
      for each kernel row $j$ do
        $j$ goes from $-(\text{height}-1)/2$ to $(\text{height}-1)/2$
\[ \Delta i = x+i-u, \quad \Delta j = y+j-v \]
\[
\begin{align*}
\text{gridXX}(x+i,y+j) & \;= \; VXX(u,v,w) \; C(\Delta i, \Delta j) \\
\text{gridXY}(x+i,y+j) & \;= \; VXY(u,v,w) \; C(\Delta i, \Delta j) \\
\text{gridYX}(x+i,y+j) & \;= \; VYX(u,v,w) \; C(\Delta i, \Delta j) \\
\text{gridYY}(x+i,y+j) & \;= \; VYY(u,v,w) \; C(\Delta i, \Delta j)
\end{align*}
\]

Note that unlike usual image processing convolutions the samples at \((u,v)\) are not on the grid points \((x,y)\) so the convolution is utilised at non-integer fractional positions \((\Delta i, \Delta j)\), complicating a matrix representation of the convolution. To avoid calculating the kernel at arbitrary fractional values during gridding each of \(\Delta i\) and \(\Delta j\) might be rounded to the nearest multiple of 0.125 (for 8 times oversampling), and 64 versions of the kernel \(C\) pre-calculated before gridding. The version of the kernel chosen for each visibility depends on the number -0.375, -0.25, -0.125, 0, 0.125, 0.25, 0.375, 0.5 to which each of \(u-x\) and \(v-y\) is closest.

### 3.1.2 Cost

The cost of gridding scales linearly with the number of visibilities, \(N_{\text{vis}}\). For convolutional gridding, each visibility is multiplied by every pixel of the convolution kernel such that the complexity of convolutional gridding is

\[
C_{\text{grid}} = O(N_{\text{vis}}N_{\text{kernel}}^2),
\]

where \(N_{\text{kernel}}\) is the kernel support size in pixels; this may include an over-sampling factor for the gridding process.

### 3.1.3 Scalability

A difficulty with parallelising this approach to gridding is that the grid accumulators will be updated by multiple visibilities, so processing visibilities in parallel requires some synchronised access to the grid accumulators. For example, in the case of SKA1-SURVEY with a sample time of at most 3s, there are at least 218880 visibilities being produced each second, each with 218 channels (across 500 MHz), so a 12 hour observation must grid at least \(2.4 \times 10^{15}\) points. If a separate \(214 \times 214\) grid is used for each 210 consecutive channels (a 2 MHz sub-band for continuum imaging) with a \(7 \times 7\) kernel size then on average each grid accumulator will be updated about 1.7 million times during gridding. Synchronising access to the grid accumulators could easily become a bottleneck, although the visibilities can be partitioned for parallel gridding on multiple copies of the grid and the results later merged (added) together into a single grid.

Gridding distributed over time is currently implemented in CASA (task `pimage`). This necessitates the creation of multiple duplicate grids, which are FFT-ed separately and the resulting images gathered and combined in the image domain.

Gridding scaled by frequency is currently implemented in ASKAPsoft. This produces a strong scaling and has the advantage that the calculation of frequency dependent kernels is naturally parallelised as well. This is currently not implemented in CASA due to non-compatibility with the existing A-projection kernel calculation.
### 3.2 W-projection

#### 3.2.1 Description

One of the main data processing challenges which appear in the wide-field imaging regime stems from the breakdown of the 2-D Fourier transform approximation of the measurement equation (relationship between measured visibilities and the sky brightness distribution). Ignoring polarisation and calibration effects, the latter can be expressed as follows for a given baseline:

\[
V(u,v) = \int \frac{I(l,m)A(l,m)}{\sqrt{1-l^2-m^2}} \exp \left\{ 2\pi j \left( ul + vm + w\sqrt{1-l^2-m^2} - 1 \right) \right\} dldm, \tag{1}
\]

where \(w\) is the baseline component towards the chosen tangent point of the image and \(A(l,m)\) is the direction-dependent instrument response (strictly speaking, this factor is different for different baselines and could also be time-variable). In this form, the measurement equation covers the following wide-field effects:

- **Non-coplanar baselines effect or \(w\)-term** (see [RD12;RD13;RD14]). The physical interpretation of the \(w\)-term problem is straightforward. When the coherence function is not sampled at the same plane, voltages have to be propagated to a common reference plane with the Fresnel propagator as this plane is located in antenna's near field.

- **Time variations of the antenna response pattern and direction-dependent propagation effects.** This includes gain changes in the system with beamformer, rotation of asymmetric beam on the sky, pointing errors and the ionospheric propagation. Frequency variations of antenna primary beam can also be taken into account via \(A(l,m)\).

The effect of both \(w\)- and \(A\)-terms can be accounted for in either image or uv-domain or in part in both domains. Theoretically, the choice or tradeoff should depend on the adopted approximations (largely about the \(A\)-term), the array configuration (magnitude of non-coplanarity) and on how well the algorithm matches the available processing hardware (largely memory and I/O constraints). For example, the time variations of \(A(l,m)\) are often ignored for interferometers with stable elements like dishes, unless a high dynamic range is required. If such an approximation is acceptable, \(A(l,m)\) is easier to take into account in the image plane post-gridding. This step can also be combined with linear mosaicing. In practice, not all possible combinations are readily enabled by the existing software and this is the area of ongoing research.

The \(w\)-term introduces a phase error which is quadratic with the distance from the tangent point. In general, it can be ignored for small Fresnel numbers

\[
R_F = \frac{\theta_{FOV}^2}{\theta_{res}} \ll 1, \tag{2}
\]

where \(\theta_{FOV}\) and \(\theta_{res}\) are the field of view and resolution, both in radians, but becomes an issue for low frequency high resolution arrays. All SKA Phase I components (LOW, MID and SURVEY) have \(R_F > 1\) (see Figure 2 in [RD15]). Below we list various approaches to correct the \(w\)-term.

The \(w\) phasor is a multiplicative term in the image domain which can be considered as a convolution in the uv-domain [RD16] with the kernel which is the Fourier transform of the
corresponding complex exponent

\[ G(u,v,w) = \int \frac{\exp\left\{2\pi j \left(w\sqrt{1-l^2-m^2}\right)\right\}}{\sqrt{1-l^2-m^2}} \exp\left\{2\pi j (ul+vm)\right\} \, dl \, dm. \] (3)

This convolution operation can be done as part of the gridding and degridding procedure by using a different kernel. In practice, the kernels are cached for a range of \( w \) and the closest matching kernel is selected for a given baseline. In this approach, the \( A \)-term can also be taken into account, including the time dependence (this is called AW-Projection), although efficient generation of such convolution functions becomes a difficult problem on its own. The main drawback of this approach is the size of convolution kernels and a poor match for modern computing architectures (the performance tends to be limited by memory bandwidth for large kernels resulting in poor scaling and bad efficiency). Note, taking the \( A \)-term into account acts as an anti-aliasing filter and regularises the kernel. An efficient implementation of this algorithm requires support for variable kernel sizes with \( w \) (and offsets for AW-Projection in the case of multi-beam instruments).

For the SKA, \( w \)-kernel support sizes are shown in Fig. 5, appropriate for truncating the kernel at the 1% level.

Figure 5: SKA1 \( w \)-kernel support sizes as a function of \( w \). The solid line refers to SKA1-LOW, the dashed line is SKA1-MID and the triple dotted-dashed line is for SKA1-SURVEY. Two cases are considered: a fiducial case (left) at a reference frequency of 150 MHz, 1 GHz and 1 GHz (low, mid and survey) as well as a worst case (right) at 70 MHz, 400 MHz and 800 MHz (low, mid and survey). The kernel size does not include oversampling (i.e. it is just the kernel width).

**Faceting** This is one of the oldest known approaches to solve the wide-field problem [RD13]. The \( w \)-term can be ignored for small enough fields of view. Therefore, the whole field can be split up into a number of facets (the exact number depends on the desired quality of the image, but it scales as \( N_{\text{facet}}^2 \)) within which the \( w \)-term can be ignored before merging them together into a final image. This method has small convolution kernels and can easily take into account the spatial variations of \( A(l,m) \) at the scale of a single facet (e.g. ionospheric correction). However, many facets are required in practice per primary beam. Therefore, this creates a non-trivial cross-talk problem and hinders efficient parallelisation. In addition, this approach tends to waste the memory in the overlap regions. It is worth noting, that this approach could be combined with other ways of treating the \( w \)-term which are discussed below. Therefore, for the multi-beam components (LOW and SURVEY), one could reduce the effective \( w \)-term by implementing one facet per beam. In this case, all drawbacks of this method would disappear.
**w-stacking**  This algorithm treats the $w$ term in the image domain post-gridding. To achieve this, one needs either multiple grids (one per $w$ plane analogous to the $w$ planes in the stack of convolution functions in the previous method) or multiple iterations over the data resulting in a 2-3 orders of magnitude larger total memory footprint or the I/O load. However, it has very small convolution kernels (just the anti-aliasing filter, which is a few uv pixels across) and can have distributed grids (although causing unbalanced loads as the number of baselines with larger $w$ is smaller).

**w-snapshots**  This approach uses the fact that for a co-planar array, i.e. $w = au + bv$, the effect of the $w$-term is equivalent to the position-dependent shift and can be accounted for by the coordinate system distortion (see [RD12;RD13;RD14;RD15]). This distortion is, in general, time-dependent and can be corrected by the image plane reprojection. Once in the common coordinate grid, individual snap-shots are accumulated. This method has an inherent error related to the slow reprojection rate. In addition, for a non-coplanar array this approach is only approximate regardless of the rate of reprojections. However, this method can be very effectively combined with the W-Projection (or W-Stacking or other related algorithms) removing this source of errors entirely [RD15]. In this case, the W-Projection sees only the residual $w$-term for a single snap-shot while the bulk of the effect is taken care of by the image-plane reprojections. This results in smaller convolution kernels than for the pure W-Projection, but still allows to achieve an arbitrary precision. The number of snap-shots (or the reprojection rate) is a free parameter which controls the magnitude of the residual $w$-term and can be optimised for. This is currently the best available approach for the baseline lengths of the order of 10-20 km. However, the benefits of this approach diminish for much longer baselines when the curvature of the Earth becomes to dominate non-coplanarity. These limitations could be alleviated somewhat by having separate grids for different groups of baselines to ensure that each such group does not deviate much from a coplanar array (although the best-fit planes and corresponding coordinate distortions are going to be different for each such group of baselines).

### 3.2.2 Cost

Calculating the $w$-kernel includes kernel generation and FFT:

$$C_{CCF,w} = 0\left( (N_{kernel}^2) + (GCFN_{kernel})^2 \log_2((GCFN_{kernel})^2) \right),$$

and we denote the support of the resulting convolution function $N_{vff} = GCFN_{kernel}$. Kernel generation is consequently dominated by the FFT.

Re-projection in the case of $w$-snapshots includes contributions from both co-ordinate generation and image re-sampling:

$$C_{re-proj} = 0\left( (N_{pix}/N_{dec})^2 + (N_{pix}^2) \right);$$

where $N_{dec}$ is a decimation factor. Re-projection is discussed further in § 3.7.

The exact form of each $w$-kernel is dependent on the individual $uvw$-value of each continuous visibility datum. Consequently, to calculate kernels individually for all visibilities would have complexity

$$C_{CCF,all-w} = N_{vis}C_{CCF,w},$$

where $N_{vis} = N_{freq}N_{time}N_{basel}$. 

---

**Document no.:** SKA-TEL-SDP-0000030  
**Revision:** 1.0  
**Release date:** 2015-02-09  
**Author:** A. Scaife  
**Page 27 of 55**
However, making a precise calculation of the convolution function support for all \(w\) values present in the observation is often not practical. Instead, the maximum \(w\) value is first determined by inspection of all the data points, and the \(w\)-spectrum is divided into \(N_{\text{planes}}\) \(w\)-planes (bins) of equidistant \(w\) intervals. All visibilities are then assigned to these bins according to their \(w\) value. All visibilities that are members of a particular bin are assumed to have the same \(w\). This reduces the number of \(w\)-kernels which need to be calculated. Hence the complexity to compute the complete set of \(w\)-kernels is given by

\[
C_{\text{CCF,all-w}} = \sum_{i=1}^{N_{\text{planes}}} (N_{\text{kernel},i}^2 + N_{\text{cvff},i}^2 \log_2 N_{\text{cvff},i}).
\]

In the case that all kernels have the same support size, this can be approximated as:

\[
C_{\text{CCF,all-w}} = N_{\text{planes}}(N_{\text{kernel}}^2 + N_{\text{cvff}}^2 \log_2 N_{\text{cvff}}^2).
\]

Gridding using the \(w\)-kernel has a complexity that is proportional to \(N_{\text{kernel}}^2\), hence the support of the \(w\)-kernel is important for determining the cost of the overall gridding process. Higher \(w\)-values (planes) require larger kernel support, \(N_{\text{kernel},i}\). The size of a kernel corresponding to some fraction, \(\eta\), of the maximum value can be found using the following approximation

\[
|u| \simeq \sqrt{(w\theta_{\text{FOV}}/2)^2 + (w\theta_{\text{FOV}}/2)/(\pi \eta / \sqrt{w})},
\]

where \(\theta_{\text{FOV}}\) is the size of the image to be constructed. Correspondence of this approximation to the full analytic treatment is shown in Fig. 6.

However, from ASKAPsoft simulations, for kernels including both \(w\)-projection and anti-aliasing, the required support is in fact smaller than that given by the previous equation. This is due to the smoothing effect of the anti-aliasing kernel on the super-Nyquist \(w\)-oscillations in the kernel tail. Indeed, multiplying \(w\theta_{\text{FOV}}\) by a constant for a given cutoff gives a reasonable match to numerical calculations. For example, \(1.12 \times w \times \theta_{\text{FOV}}\) for a kernel with a cutoff of 0.1\%. This is shown empirically for kernels generated in ASKAPsoft using both a \(w\)-component and an anti-aliasing component in Fig. 7. The inclusion of an \(\lambda\)-component in the combined kernel also has a similar effect, see §3.3.2. Further investigation shows that this behaviour is reasonably general when considered for situations other than an SKA1-SURVEY-like instrument and this will be verified as part of ongoing work and the combined gridding kernel size amended appropriately.

Consequently the cost of gridding visibilities with large-\(w\) is more expensive than those with small \(w\)-values and can dominate the computational load. If small reductions in dynamic range can be made it is possible to reduce the compute cost of the \(w\)-projection gridding significantly. Specifically, tests show that for SKA1-MID simulations, discarding \(2-10\%\) of the data with the highest \(w\)-values can reduce the computational cost by \(40-80\%\) at the cost of \(3-20\%\) of the dynamic range. These results are summarised in Table 2.

<table>
<thead>
<tr>
<th>% Visibilities Discarded</th>
<th>% Reduction in Computational Cost</th>
<th>% Reduction in Dynamic Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>42.6</td>
<td>3.0</td>
</tr>
<tr>
<td>5</td>
<td>63.2</td>
<td>8.3</td>
</tr>
<tr>
<td>10</td>
<td>86.3</td>
<td>18.9</td>
</tr>
</tbody>
</table>

Table 2: Dynamic range vs. computation saved when removing highest \(w\)-points, for SKA1-MID using uniform weighting.

Unrestricted
Author: A. Scaife
Page 28 of 55
Figure 6: $w$-kernel function for an SKA1-Survey-like array at maximum $w$ ($1.33 \times 10^5 \lambda$).
Figure 7: ASKAPsoft $w$-projection kernel size. Points indicate the size used in Cimager, while solid lines indicate the nominal kernel size, $w_{\text{FOV}}$. Setting a reasonable threshold will increase kernel size beyond $w_{\text{FOV}}$, however convolution with the anti-aliasing function significantly decreases the wings of the $w$-projection function and limits the increase. Plateaus indicate an insufficient maximum support size.

3.2.3 Scalability

- $w$-stacking can have application of the measurement operator distributed by $w$-plane.
- $w$-snapshots can have application of the measurement operator distributed in time.
- Faceting allows processing to be distributed by facet. There is no reduction in the input data size, but $uv$-grids and images can have smaller $N_{\text{pix}}$ due to the reduced FOV per facet.
- It is also possible to distribute in frequency. A timing comparison of SKA1-scale simulations for $w$-projection performed in series and in parallel is shown in Fig. 8.

3.3 A-projection

3.3.1 Description

Observations made with radio interferometers suffer from variable gain effects that are broadly classified as direction independent (DI) and direction dependent (DD) errors. DI gain errors are independent of direction (hence the name) and result from the electronics of the system and are the classic gain errors that are solved for during phase and amplitude self-calibration. DD gain errors (or DDEs) are time-varying gains that result from asymmetries of the primary beam and the atmosphere (largely the ionosphere at SKA frequencies).

The antenna response, $A$, or the primary beam, is the Fourier transform of the autocorrelation of the complex voltage distribution of the field, $f(u,v)$ in the aperture of the antenna [RD17;RD18]. For parabolas (or dishes) $f(u,v)$ can be described by the antenna illumination function. Thus, the form of $A$ is dependent on the way in which the feed illuminates the aperture. A perfectly illuminated, unblocked parabola can be represented by a top-hat function, for which the Fourier Transform is a sinc (or Bessel) function. Thus resulting in a symmetric primary beam. However, in practice this is not the case and the primary beam is asymmetric due to three main effects:
Figure 8: Cimager imaging times for 10 km-array visibilities processed with various parameters. Points show measured data and lines show modelled data. To limit the size of $w$-projection kernels, visibilities were divided in time into snapshots that were imaged separately and combined after image re-sampling. The horizontal axis indicates the largest $w$ value allowed per snapshot, given as a percentage of the maximum $w$ value, and there are a proportional number of $w$-planes (i.e. constant $w$ sampling). Starred data points indicate Cimager with MPI enabled, and highlight processing savings associated with cached gridding kernels. Labels M, dec and sup represent $N_{\text{pix}}$, $N_{\text{dec}}$ and $(Q_{\text{GCD}}N_{\text{kernel}})$, respectively.

1. Partial blockage of the collecting area by the feed and the feed-supports;
2. Offset between the polarisation feeds (known as beam-squint);
3. Pointing errors.

In addition to these instrument-based terms an additional direction-dependent amplitude and phase contribution is also caused by the Earth’s ionosphere. This can be incorporated into the same $\Lambda$-term.

This asymmetric beam will rotate on the sky as a function of parallactic angle during an observation, creating a time and direction-dependent effect, which must be accounted for in order to form high dynamic range images. Since it has components for each polarisation feed on each dish/antenna the full effect for each visibility is represented by a Mueller ($4 \times 4$) matrix, the anti-diagonal terms of which are about 6-7 orders of magnitude lower than the diagonal, while the off-diagonal terms are 2-4 orders of magnitude lower. Thus, for high dynamic-range imaging in Stokes-I and full-polarisation imaging (Stokes-Q, -U and -V), with the SKA, it is anticipated that the off-diagonal terms will be required.

Ionospheric phase screens are caused by variations in the total electron content (TEC) of the ionosphere. This varies as a function of both time and position and the phase effect varies as a function of frequency, being more significant at lower frequency. Input models of the TEC content in the ionosphere can be provided by GPS systems and additional refinement of these models can be made during self-calibration. Typical time-scales on which substantial variations are present in the ionosphere vary from 10s of seconds to 10s of minutes, depending on
the time of day/night. A-kernel update rates should reflect these changes as well as those of the asymmetric beam model.

To correct for these effects during imaging, the A-projection method resamples the observed data, $V^\text{Obs}_k$, onto a regular grid using a convolution function, adj($A^M_k$), that is derived from a model of the primary beam, $A^M_k$. The resulting gridded data is accumulated in the data (i.e. visibility) domain and Fourier transformed to compute the continuum image, which is then normalised by the accumulation of the primary beam model to recover an absolute scaling.

The A-projection method corrects for DD errors associated with the primary beam in two fundamental steps:

1. The observed data, $V^\text{Obs}_k$ are resampled onto a regular grid using a convolution function, adj($A^M_k$) that is derived from the primary beam and ionosphere model, $A^M_k$. The resulting gridded data is accumulated in the data (i.e. visibility) domain and then Fourier transformed to compute the continuum image;

2. The primary beam model, $A^M_k$ is accumulated over all values of $k$ and is used to scale the resulting image by dividing out the model primary beam accumulation.

This convolutional gridding process can be combined with that of w-projection, resulting in the so-called “AW-projection”. If an ionospheric phase screen is included in the A-part of the kernel, it is also sometimes called “IAW-projection”.

### 3.3.2 Cost

Critical parameters for the A-projection computational cost are:

1. The rate at which the kernel is updated, described as a function of time or parallactic angle;

2. The support size of the kernel in pixels.

The first of these is linked to the achievable dynamic range of the final image. For the EVLA beams, the relationship between dynamic range (DR) and A-projection kernel update rate is shown for high DR in the Fig. 9. The same relationship is shown for SKA1-MID based on the DVA1 beam [RD31] and example array geometries.

For required SKA1 DR values, this implies A-projection kernel updates for parallactic angle increments of $<< 1$ degree. However, this rate is dependent on the exact beam shape and will need to be calculated for the SKA based on updated beam models for both dish and aperture array based instruments.

The support size of the kernel in pixels, $N_{\text{kernel}}$, is determined by the pixel-size required for the kernel and the size of the kernel itself. The pixel-size is determined by the size of the dish/station and the size of the kernel is determined by the highest spatial frequency component of the beam, which is the beam-squint. Representative values for the EVLA and SKA1 dish arrays, using the SKA-DVA1 beam, are shown in Table 3.

When combining with the w-kernel, the tail of the w-function comes from averaging over w-oscillations above the Nyquist frequency. Convolving with an anti-aliasing function or an A-projection function averages these further and reduces the width at a given cutoff level. Consequently, blindly estimating the w-kernel size as described in §3.2.2 and then adding the A-kernel will give an over-estimate of the support required for the combined gridding kernel. Including an A-kernel will reduce the predicted size of the combined kernel, as well as minimizing the need for an anti-aliasing term as this is also inherently contained in the A-kernel itself.
Figure 9: The expected dynamic range plotted against the corresponding parallactic angle increment for the VLA (blue) and SKA1-MID (red).

Table 3: A-kernel parameters for various arrays

<table>
<thead>
<tr>
<th>Array</th>
<th>$\lambda$ (m)</th>
<th>$D_A$ (m)</th>
<th>$(2.D_A)/\lambda$</th>
<th>$E_{\theta_{bs}}$ (arcsec)</th>
<th>$C_F$ size (k$\lambda$)</th>
<th>$N_{kernel}$ (pixels)</th>
</tr>
</thead>
<tbody>
<tr>
<td>EVLA</td>
<td>0.92</td>
<td>25</td>
<td>54</td>
<td>218.6</td>
<td>0.9</td>
<td>17</td>
</tr>
<tr>
<td></td>
<td>0.21</td>
<td>25</td>
<td>238</td>
<td>49.9</td>
<td>4.1</td>
<td>17</td>
</tr>
<tr>
<td></td>
<td>0.06</td>
<td>25</td>
<td>833</td>
<td>14.3</td>
<td>14.5</td>
<td>17</td>
</tr>
<tr>
<td></td>
<td>0.03</td>
<td>25</td>
<td>1666</td>
<td>7.1</td>
<td>29.0</td>
<td>17</td>
</tr>
<tr>
<td>SKA Dish</td>
<td>0.92</td>
<td>15</td>
<td>33</td>
<td>$\leq 632.5$</td>
<td>$\geq 0.3$</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>0.21</td>
<td>15</td>
<td>143</td>
<td>$\leq 144.4$</td>
<td>$\geq 1.4$</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>0.06</td>
<td>15</td>
<td>500</td>
<td>$\leq 41.3$</td>
<td>$\geq 5.0$</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>0.03</td>
<td>15</td>
<td>1000</td>
<td>$\leq 20.6$</td>
<td>$\geq 10$</td>
<td>$\geq 10$</td>
</tr>
</tbody>
</table>

Notes.
* Considering only the DVA1 antenna, and applicable to both Mid and Survey
† $N_{kernel}$ is set to 16 pixels in CASA for the EVLA. This would cause an undersampling of the beam squint by approximately 6%.
(1) Array names; (2) Observing wavelengths; (3) Antenna/station diameter;
(4) $1/E_{\theta_{bs}}$ (see text); (5) For dishes, this is the beam squint: the JVLA, $\theta_{bs} = \pm 237.56\lambda$(arcsec/m); for the DVA-1 $\theta_{bs} \leq 0.05$ HPBW [RD31]; (6) Size of the convolution function = $1/\theta_{bs}$; (7) Size of the convolution function in units of pixels.

Note: From one Major Cycle to the other, the only change is with the model image, $I^M$. Since $A_k$ does not change, both the convolution functions, $(A_k^M)^\dagger$ and $A^M$ can be cached for use in subsequent Major Cycles.
Assuming an element of the Mueller matrix has $N_A$ pixels, the computational cost of calculating the convolution kernel for each parallactic increment on each baseline independently will be the cost of rotating (re-projecting) the antenna power pattern, FFT, followed by calculation of the adjoint:

$$C_{\text{CCF}, A} \propto C_{\text{rot}} + C_{\text{FFT}} + C_{\text{adj}}.$$ 

The total complexity will therefore be

$$C_{\text{CCF}, A} = 0(N_{\text{inc}} \times N_{\text{basel}} \times N_{\text{mm}} \times (N_A^2 + N_A \log_2 N_A + N_A^2))$$

where $N_{\text{mm}}$ is a factor to account for the extra operations required for including off-diagonal elements of the Mueller matrix. This calculation needs only be done once, if there is sufficient memory available to store the kernels.

### 3.3.3 Scalability

Computation of the A-projection kernel scales naturally in time and frequency.

### 3.4 Fast Fourier Transform

#### 3.4.1 Description

The naïve approach to calculating the discrete Fourier transform as a matrix multiplication is an $O(N^4)$ algorithm, but in 1965 the Fast Fourier transform was (re)discovered by Cooley and Tukey [RD19] (although already known to Gauss around 1805), as an $O(N \log_2 N)$ divide-and-conquer approach to computing the matrix multiplication.

The most common form of the Fast Fourier transform is the radix-2 case, where the input data are combined together pairwise at a time using butterfly operations in $\log(N)$ rounds. Each of the $N \log_2(N)$ butterfly operations requires one multiplication, one addition and one subtraction. For complex input each complex addition/subtraction is two real additions. By default, a complex multiplication is four real multiplications and two real additions but Gauss's complex multiplication algorithm replaces this by three real multiplications and three additions (possibly an improvement particularly on older systems where multiplications were often more computationally demanding than additions). In total this is $10 \frac{N^2}{2} \log_2(N)$ real operations for an $N$-point radix-2 FFT (without including any window function which requires a further $N$ multiplications, or any polyphase filter).

There are two alternative ways the FFT is implemented, although both perform equivalent operations on the input data and give identical output. (1) Using a decimation in time the data are initially rearranged using a bit-reversal and then the ‘butterflies’ are first applied to consecutive data values ($m/2 = 1$), then with a skip of $m/2 = 2$ etc, with a final round using skip $m/2 = N/2$; (2) using a decimation in frequency a butterfly operation multiplies by the complex root of unity $\chi$ after the addition and subtraction, and the butterflies are first applied with a skip $m/2 = N/2$, then with half that skip $m/2 = N/4$, etc, until the final round using skip $m/2 = 1$, and a bit reversal used on the output data to correctly reorder it.

#### 3.4.2 Cost

The Fast Fourier Transform has complexity of

$$0(N_{\text{pix}}^2 \log_2(N_{\text{pix}}^2)),$$
where a pre-factor is often stated, however, the exact value of this pre-factor varies depending on

- Implementation and details of the algorithm;
- Radices (prime or composite).

Both CPU and GPU FFT efficiency are found to be approximately $8 - 15\%$ of peak; it should be noted that

- CPU efficiency depends also on the number of threads/cores: the larger the number of threads, the lower the efficiency for the same memory bandwidth;
- GPU efficiency depends on the initial and final data locations. If data need to be transferred to the GPU and results collected from it, the very low ratio between number of Flops and I/O data would increase considerably execution times.

Internal I/O requirements also depend on the implementation, bearing in mind that each radix would require a complete refresh (i.e. read/write memory access) to $2N^2$ data items. So, large composite radices can be used to reduce memory refresh, as in the case of cuFFT. Hence, internal I/O cannot be modelled here.

For example, for a K40 GPU, capable of 1400 GFlops/sec in double precision, for $N = 16384$ and maximum PCI-e bandwidth $B_p=16$ GBytes/sec we would have a computing time $T \sim (5 \cdot 16384^2 \cdot 14)/(0.1 \cdot 1.4 \cdot 10^{12}) \approx 0.13$ seconds. However, data transfer would require well over 0.5 seconds to transfer input and output data through PCI-e.

Empirical results for FFT run-time tests can be found in § 3.5, Fig. 10.

### 3.4.3 Scalability

As $N_{\text{pix}}$ grows so does the arithmetic intensity of the FFT, so it becomes advantageous to use a multi-step FFT, both to parallelise the butterfly operations for a multi-core processor and to help keep the block sizes within the processor cache size for performance.

The SKA-TEL.CSP.CBF.SUR Sub-element Prototype Test Report Part 2 Many Core FFT demonstrates how a six-step FFT is used to achieve a 1D 218 point FFT entirely on-chip within a cluster of 16 cores, exploiting processor local cache, and generalisable to a 2D $2048 \times 2048$ point FFT entirely on chip. By 2015 a 2D $4096 \times 4096$ point FFT should also be achievable entirely on chip. When $N_{\text{pix}}^2$ is too large for the data to fit within processor memory a significant performance and power cost is incurred as the data must be transferred on and off chip multiple times, but a six-step FFT can still be used to reduce the repeated I/O with external memory.

For multidimensional FFT, such as the 2D (inverse) FFT for the Imaging Pipeline there are further considerations. The split-radix approach for 1D FFT, which uses a combination of radix-2 and radix-4 butterfly operations, can be generalised for multidimensional FFTs to vector-radix algorithms. The paper shows how 2D vector-radix algorithms decreases the overall number of arithmetic operations compared to a radix-4 implementation by about 5%.

If data for a 2D FFT is too large to fit within processor memory, then the data order in external memory should be considered to avoid to penalties incurred with strided access to off-chip memory. If 2D data were simply stored row by row in off-chip memory, and a 2D FFT applied by applying a 1D FFT to each row followed by (or preceded by) a 1D FFT to each column, then the FFTs applied to the rows would process a contiguous block of data each time, so would make good use of processor caching and block-wise data transfers, but the
FFTs applied to the columns would all use strided memory access, incurring poor bandwidth and higher power consumption. Instead, for a large 2D FFT the data can be stored in memory to avoid strided memory access by partitioning the 2D data into tile blocks, with the data from each tile stored consecutively in memory. The use of tiles for 2D and 3D FFTs with DRAM is described in detail in [RD20] and incorporated within Spiral FFT implementations.

3.5 Sparse Fourier Transform

3.5.1 Description

Sparse Fourier Transforms (SFTs) are algorithms designed to deal specifically with the sparse Fourier transform problem: given a complex vector $x$ of length $n$, estimate the $k$ largest (in magnitude) coefficients of the conjugate Fourier space $\hat{x}$. Unlike DFT algorithms designed to recover the complete $\hat{x}$ where runtime is $O(n)$, one can design algorithms specifically for the case where $\hat{x}$ is sparse, i.e. $k << n$, which have an output sensitive runtime dependent on $k$. Traditionally such algorithms have been constructed to satisfy the $\ell_2/\ell_2$ guarantee:

$$||\hat{x} - \hat{x}'||_2 \leq C \min ||\hat{x} - y||_2,$$

where $C$ is an approximation factor.

Image enhancement for SFT reconstructed images The sparse FFT (SFT) algorithm produces as its output a single data file containing the intensities of each pixel stored in complex double type format. We would normally expect the majority (> 95%) of these pixels to have a zero value, as the algorithm will focus its efforts only upon those regions of the output image in which sources are detected. To construct an image from these data requires the insertion of a grey-scale bitmap (BMP) file header, and the normalisation of each pixel onto a single-byte scale (256 intensity levels). The choice of black and white level for this normalisation (i.e. the values from the SFT output that are mapped to image intensity 0 and 255 respectively) is of critical importance to the quality of the final images, and should be chosen in order to eliminate as much of the noise as possible.

Once data are in image form, reducing the noise requires the application of image processing techniques, and choosing an appropriate strategy is crucial. For example, techniques for removing cosmic rays from Hubble Space Telescope (HST) data bear similarity to those used to remove ‘silvering’ from old black-and-white photographs, in which two separate images of the same scene are combined using a suitable blending mode such as ‘darken only’. Since the unwanted artifacts appear at random locations, any bright spots appearing in only one input image are removed from the combined image, and for HST data the quality of the result far exceeds that which could have been achieved by capturing one single long exposure image. SDP tests show that the distribution of noise intensity in SFT reconstructed images using VLA data takes the form of a gamma function, with shape parameter $k = 11$ and scale parameter $\theta = 1.588 \times 10^{-4}$. Satisfactory image quality could be achieved by setting the black level to be just above the peak in the gamma distribution, effectively replacing most of the noise with a black background. This simple adjustment to the pixel normalisation could be sufficient for SKA requirements and is the recommended approach.

However, to enhance images with particularly faint sources (i.e. 2 or 3 sigma) it may be possible to blend images constructed from two (or more) sequential time periods in a similar way to that discussed above for HST. This technique would cause the peak in the noise distribution to narrow, thereby enhancing (relative to the noise) any pixel, which maintained
its value across multiple time periods. This enhancement, however, would only be effective if a transient source continued to change in brightness across multiple time periods, and further testing is required to determine under what circumstances, this technique is preferable to simply constructing a single SFT image from the individual time periods.

### 3.5.2 Cost

Historically, SFT algorithms have a runtime which is polynomial in $k$ and $\log n$, with the fastest variants having run times of $O(k^a \log^b n)$ (refs.) where $a = 1, 2$ and $b > 2$. This has limited their usefulness, above a standard FFT to situations with $n \geq 2^{22}$ and $k \leq 135$.

A more recent implementation of SFT [RD21] has been made, with a simplified structure which has a theoretical runtime of

$$O(\log N_{\text{pix}}^2 \sqrt{N_{\text{pix}}^2 k \log N_{\text{pix}}^2}),$$

and it is a generalised implementation of this algorithm, $\text{slabFFT}$, which has been the basis of our tests.

In all cases, the performance of sparse methods for Fourier transforming astronomical images depends not just upon the number of sources in the image but also upon the number of pixels which are occupied by those sources. Calculating this sparsity, $k$, exactly for images where structure is convolved with extended functions is non-trivial; however, it is clear that the width of the point spread function becomes increasingly important as image sizes are reduced. Therefore, tests in which the image field of view (FOV) is kept constant as the total number of pixels ($N_{\text{pix}}^2$) varies will have different performance characteristics from tests in which the angular size of each pixel is kept constant.

We summarise here the observed performance characteristics of the GPU implementation of $\text{slabFFT}$; the CPU version scales almost identically. For tests in which the FOV is kept fixed (such that the fraction of the image occupied by the sources remains approximately constant) we find that the operation count scales as

$$C_{\text{slabFFT}, N_{\text{pix}}} = 0(N_{\text{pix}}^{1.5} \log_2 N_{\text{pix}}^2),$$

and for tests in which the image size in pixels is held constant while the number of sources is increased, we find that

$$C_{\text{slabFFT}, k} = 0(k^{0.5}).$$

These results agree approximately with the theoretical scaling. In both cases the CPU utilisation was between 8-12%. The results of these tests are illustrated in Fig. 10.

The number of operations also scales with the ‘threshold value’ ($V_t$), which sets the minimum ‘low-resolution’ pixel intensity for which the algorithm will identify and populate the corresponding pixels in the high-resolution output image; pixels below the threshold value are set to zero in the output image. The operation count scales roughly as

$$C_{\text{slabFFT}, V_t} = O(V_t^{-3.4});$$

$V_t$ should be carefully chosen (and adjusted) to balance performance and image quality during implementation of the algorithm. Further verification tests are underway to calibrate this w.r.t. defined detection requirements.
3.5.3 Scalability

The cuSlabFFT code is highly parallel and further improvements to the GPU-based implementation are being developed in collaboration with Nvidia and MIT, which are expected to result in further runtime improvements of \( \sim 10 \times \) on GPU.

3.6 Major Cycle
3.6.1 Description

The incomplete Fourier sampling produced by an interferometric observation results in a logical multiplication of the true visibilities in the Fourier plane. Consequently, when these visibilities are Fourier transformed into image space they are convolved with a function that can be calculated as the Fourier transform of the logical sampling pattern, known as the dirty beam. This convolution can have the effect of obscuring and confusing structures in the image plane, which would otherwise be visible/distinct. In order to account for this convolution radio images are subjected to a deconvolution process to remove the effect of the dirty beam. Deconvolved images are known as clean images as opposed to dirty images. (Note: this nomenclature persists for deconvolved images not reconstructed using a CLEAN algorithm). Typically, a clean image will be reconstructed by convolving the recovered image components with a Gaussian clean beam fitted to the main lobe of the dirty beam. This reconstruction is done to avoid over-interpretation of the final images during scientific analysis, as the image components, known as clean components, are generally identified on scales smaller than the true resolution of the instrument. This process produces a number of outputs. For a monochromatic dataset (i.e. single frequency) these are:

1. An image of the dirty beam. Dirty beams can contain complex structure and are typically not readily expressible in simple analytic form.
2. An image, or list, of the clean components. These are typically expressed in some pre-defined basis, with a specified location and amplitude.

3. A clean image of the target field. This will have been reconstructed using the clean beam.

4. Dimensions for the clean beam, generally expressed as (major axis, minor axis, position angle).

For multi-frequency data, where a Taylor expansion in frequency is typically used to describe the frequency dependence of individual clean components, a number of additional products are also produced:

5. A spectral index clean image (first moment)

6. A spectral curvature clean image (second moment; if required)

In addition, the completeness of deconvolution and data quality are often assessed using

7. A residual image (or images for multi-moment deconvolution).

Here we concentrate on two deconvolution methods down-selected as appropriate for SKA.

**MS-MFS** CLEAN algorithms are the most well established method employed with radio interferometric observations for deconvolution and imaging. In its most fundamental form this algorithm represents the iterative subtraction from the dirty image of point sources convolved with the point spread function (dirty beam) in image space, known as Högbom CLEAN (Högbom 1974). This is broadly analogous to the process known in signal-processing terms as matching pursuit. A number of CLEAN-based algorithms now exist that represent adaptations from this original Högbom form. These are more commonly utilised than the original form, and incorporate multi-scale and multi-frequency synthesis algorithms (e.g. MS-CLEAN & ASP-CLEAN). The multi-scale multi-frequency-synthesis (MS-MFS) CLEAN algorithm is described fully in [RD22]. MS-MFS CLEAN follows the standard Major/Minor Cycle CLEAN iterative procedure utilising a multi-scale image model and a Taylor expansion representation of the frequency coverage.

The flow diagram for this algorithm is shown in Fig. 11. This algorithm is currently implemented within the CASA software package. Fig. 11 shows that a potential parallelisation/distribution approach for use with the predicted SKA computing infrastructure which would involve performing the initial application of the adjoint operator (i.e. the gridding and FFT) on each frequency channel across multiple compute nodes.

Within the current implementation both $A$- and $w$-projection gridding variants can be used with the MS-MFS algorithm to perform wide-field imaging. Note, Fig. 11 does not include any re-projection or instances of imaging multiple facets.

**Compressed Sensing** Compressive Sensing / Compressive Sampling / Compressed Sensing (CS) algorithms are in their infancy in comparison to CLEAN-based methods for use in radio interferometric imaging and deconvolution. However, a number of CS algorithms have now been developed for this purpose. These are fundamentally different from those considered CLEAN-based, though in their simplest form operate in a very similar way, in that a point source is located, convolved with the dirty beam, subtracted and the residuals are updated.
Compressed sensing algorithms enable the recovery of the observed signal under the condition that it exhibits sparsity in a certain basis. For a general introduction to compressive sensing and sparsity please see [RD23;RD24]. In the case of a basic imaging algorithm, the measurement model i.e. the visibilities, can be represented as $y = \Phi x + n$, where $n$ is the observation noise and $x$ represents the image (concatenated into a vector). $\Phi$ denotes the measurement operator, which in standard imaging procedures represents the transform from the visibility to the image plane.

The image $(x)$ can be reconstructed from the visibilities $(y)$ by solving the following unconstrained convex optimisation problem [RD25]

$$\min_{\bar{x} \in \mathbb{C}^N} (\lambda \|\bar{x}\|_1 + \|y - \Phi \bar{x}\|_2^2).$$  \hspace{1cm} (5)

In this context CLEAN algorithms can be considered to be examples of the Matching Pursuit algorithm (ref) used to minimise Eqn. 5. CLEAN implicitly includes a sparsity prior by utilising the assumption that the field contains only point sources (i.e. a Dirac basis).

This approach makes use of a regularisation parameter that enforces sparsity, $(\lambda)$. A number of methods have been suggested for determining this value, though the necessarily ad-hoc manner for setting $\lambda$ highlights the limitation of this approach.

Image reconstruction can also be performed by instead solving a constrained convex optimisation problem [RD26]:

$$\min_{\bar{x} \in \mathbb{C}^N} \|\bar{x}\|_1 \text{ subject to } \|y - \Phi \bar{x}\|_2^2 \leq \varepsilon.$$  \hspace{1cm} (6)

where $\varepsilon$ is an upper bound on the $\ell_2$ norm of the noise and $\| \cdot \|_1$ denotes the $\ell_1$ norm of a complex-valued vector. This effectively defines the iterative procedure of the basic form where $y - \Phi \bar{x}$ gives the residuals that will be updated following each cycle.

Alternatively, a more standard approach involves the use of wavelets to describe the image i.e. $x = \Psi \alpha$ where $\Psi$ is a wavelet transform (e.g. a Fourier basis function) and $\alpha$ are the wavelet coefficients. Eqn. 6 then becomes

$$\min_{\bar{\alpha} \in \mathbb{C}^N} \|\bar{\alpha}\|_1 \text{ subject to } \|y - \Phi \Psi \bar{\alpha}\|_2^2 \leq \varepsilon.$$  \hspace{1cm} (7)

In the synthesis setting (i.e. the standard CS setting), with each Major Cycle $\alpha$ is updated. Then once the final $\alpha$ is determined the image is computed by $x = \Psi \alpha$. 

---

**Figure 11:** MS-MFS Major Cycle Flow Diagram.
In the analysis setting, one recovers $x$ directly. This has the advantage that if you consider complicated wavelet representations, i.e. including many different wavelet types, the dimension of the space does not grow, i.e. you always recover $N_{\text{pix}}$ pixels. This is the approach of the SARA algorithm (REF).

Douglas-Rachford [RD26] and other evaluation algorithms can be applied to solve both the analysis and synthesis settings. The wavelet transform operator is tight framed and the proximity operator for $\|\bar{\alpha}\|_1$ can therefore be evaluated in a single step; for the analysis setting the update must be calculated iteratively. Iterations involve applying the frame operator (e.g. wavelet transform) and not the measurement operator, so even if iterations are required this is relatively fast.

The measurement operator is not tight-framed and consequently the proximity operator for $\|y - \Phi \Psi \bar{\alpha}\|_2^2 \leq \epsilon$ will require an iterative solution and the application of the measurement operator in both the forward and backward direction for each iteration.

A simple functional diagram representing a generic CS algorithm using the Douglas-Rachford method is shown in Fig. 12.

![Figure 12: Compressed Sensing Major Cycle Flow Diagram.](image)

### 3.6.2 Cost

**MS-MFS** The MS-MFS Major Cycle computation can be viewed as an independent application of the adjoint measurement operator for each frequency channel, followed by a reduction step to sum the contributions over frequency channels. The Minor Cycle computation then acts on the residual image and its multi-scale Taylor expansion.

The complexity of a Major Cycle is given by

$$C_{\text{major}} = 2N_{\text{freq}}(C_{\text{grid}} + C_{\text{FFT}}) + N_tC_{\text{taylor}} + NS_NC_{\text{scale}} + C_{\text{sub}}.$$  

Application of the adjoint measurement operator has the complexity,

$$(C_{\text{grid}} + C_{\text{FFT}}) = 0(N_{\text{vis}}N_{\text{kernel}}^2 + N_{\text{pix}}^2 \log_2(N_{\text{pix}}^2)).$$

The exact form of the gridding complexity depends sensitively on the implementation of any visibility weighting: in the simplest case where inverse variance weighting is used, $C_{\text{grid}} =$
0(\(N_{\text{vis}} + N_{\text{vis}} N_{\text{kernel}}^2\)); however, if a Weiner filtering approach is used (cf. ASKAPsoft:[RD27]) then the weights must be computed using a Fourier transform of the weights to compute the \(uv\)-weighting function, \(W_{\text{weiner}}\), for a given value of the Robust parameter, \(R\),

\[
W_{ij,\text{weiner}} = F \left( \frac{F^{-1}(w_{ij}^*)}{F^{-1}(w_{ij}^*)F^{-1}(w_{ij}) + 10^4 R} \right).
\]

Although this weighting scheme adds a complexity of \(\sim 0(\(N_{\text{vis}} + N_{\text{vis}} N_{\text{kernel}}^2 + 2N_{\text{pix}}^2 \log_2 N_{\text{pix}}^2\))\) (multiplication by inverse variance; convolutional gridding; calculation of Weiner filter), it does not necessarily need to be re-computed every Major Cycle, but can be done once and cached. In principle, this weighting may also contain a term dependent on the primary beam, but this is not yet implemented in ASKAPsoft.

The calculation of the Taylor term images requires the weighted sum of the \(N_{\text{freq}}\) dirty images at each pixel:

\[
C_{\text{taylor}} = 0(2N_{\text{pix}}^2).
\]

The factor 2 here accounts for the weighting of individual images before summation. If summation involves reduction across compute island this factor will increase further.

The formation of the dirty scale images requires the convolution of each Taylor term image with each scale kernel,

\[
C_{\text{scale}} = 0(N_{\text{pix}}^2 N_{\text{S}}^2);
\]

while the subtraction step operates on the de-gridded visibilities:

\[
C_{\text{sub}} = 0(N_{\text{vis}}).
\]

Since the Major Cycle is repeated across all polarisations and all beams, the overall complexity (per Major Cycle) is

\[
C_{\text{major}} = N_{\text{pol}} N_{\text{beam}} \times \left[ 2N_{\text{freq}}(N_{\text{vis}} N_{\text{kernel}}^2 + N_{\text{pix}}^2 \log_2 (N_{\text{pix}}^2)) + 2N_{\text{S}} N_{\text{pix}}^2 + N_{\text{S}} N_{\text{pol}} N_{\text{S}} N_{\text{vis}} N_{\text{pix}}^2 N_{\text{S}}^2 + N_{\text{vis}} \right].
\]

Hence, the Major Cycle is heavily dominated by the application of the (adjoint) measurement operator.

During the Minor Cycle, a new sky model component is firstly reconstructed from the dirty image residual. This is achieved by solving the normal equation in the image domain. A diagonal approximation of the composite normal equation matrix is made such that this normal equation can be decomposed into separate equations over scales and solved trivially. Secondly, the image models and residuals are updated. The asymptotic complexity of the Minor Cycle computation is

\[
C_{\text{minor}} = N_{\text{S}} N_{\text{pix}}^2 N_{\text{S}} + N_{\text{S}} N_{\text{S}}^2 N_{\text{vis}}^2.
\]

The computational load of Minor Cycle is significantly less than that of the Major Cycle.

**Compressed Sensing** The Major Cycle complexity of the constrained compressed sensing problem is similar to that of MS-MFS. A variety of different solvers exist which can be applied to calculate the proximity operators, however, the calculation of the \(\ell_2\) proximity operator (denoted L2 in Fig. 12) requires application of the measurement operator in both the forward and backward direction. The general form of the complexity of the CS Major Cycle is

\[
C_{\text{major}} = 2C_{\ell_2} + C_{\text{update}} + C_{\text{sub}},
\]
where

$$C_{L2} = 0(N_{vis}N_{kernel}^2 + N_{pix}^2 \log_2 N_{pix}^2)$$

and

$$C_{sub'} = C_{update} = 0(N_{pix}^2),$$

as both the update and the comparison (sub’) step are performed in the image plane. Note that the $\varepsilon$ based $\ell_2$ constraint is applied as an indicator function in the calculation of the proximity operator.

The Minor Cycle operates only in the image plane. Assuming an analysis setting, the frame operator (wavelet transform) is applied twice giving an approximate complexity of

$$C_{minor} = 0(2N_{pix}^2 \log_2 N_{pix}^2).$$

This makes the total complexity of the compressed sensing operation, $C_{major} + C_{minor'}$:

$$0(2N_{L2}^2(N_{vis}N_{kernel}^2 + N_{pix}^2 \log_2 N_{pix}^2) + 2N_{pix}^2 + 2N_{L1}N_{pix}^2 \log_2 N_{pix}^2),$$

where $N_{L2}$ and $N_{L1}$ are the number of internal iterations for the $\ell_2$ and $\ell_1$ proximity operators. In the case of a tight-framed operator for the $\ell_1$ constraint, $N_{L1} = 1$.

**Comparison** Consequently, the complexity per Major Cycle for both the MS-MFS and Compressed Sensing methods is similar and dominated by application of the measurement operator in the Major Cycle. However, we note that currently available implementations of the Compressed Sensing method require $\sim 10$ times more Major Cycles to reach convergence than MS-MFS. Run purely in series, this makes the Compressed Sensing method more expensive.

### 3.6.3 Memory Requirements

**MS-MFS** During the MS-MFS Major Cycle a number of different types of memory are required. Continuous visibility data, convolution kernels and the model will need to be buffered. $uv$-grids require fast access for Fourier transforms and will be held in working memory. Additional data grids, including the multi-scale Taylor term images used in the Minor Cycle will need to be held in pool memory.

**CS** During the CS Major Cycle continuous visibility data, convolution kernels and the global model will need to be buffered. Working memory will be required for the $uv$ grids in the $L_2$ proximity operator calculation and pool memory for the local model in the $L_1$ and $L_2$ proximity operator calculations.

### 3.6.4 Scalability

A naïve data distribution diagram for the overall Imaging Pipeline might look something like that shown in Fig. 1. In this scenario individual beams are processed separately. Data are then distributed in time for calculation of the $w$- and $A$-kernel components, which can subsequently be scaled as a function of frequency. For a pure $w$-projection or $w$-stacking implementation the data may then optionally be gathered in time before being distributed again in frequency. For a $w$-snapshots implementation, the distribution in time would be propagated forward. For gridding including $A$-projection, although the distribution is shown in Stokes parameter, all
polarisation products are required in the gridding step with appropriate Mueller matrix weights in the $A$-kernel.

Fig. 1 shows a generalised distribution scheme with a simplified deconvolution process, e.g. Högbom CLEAN. However, the exact action of the Major Cycle and the gather steps within it can be generalised to other deconvolution Major Cycles. In the following we expand on distribution schemes for MS-MFS and CS deconvolution methods.

**MS-MFS** The iterative nature of the MS-MFS Major Cycle does not naturally lend itself to distribution. However, certain elements within the Major Cycle may be distributed. The most obvious steps for useful distribution are the combined gridding/de-gridding and FFT stage, which operates separately on individual frequency channels, and potentially the Minor Cycle step where the model update is found. This step operates separately on both scales and pixels and, for large images, may be beneficial to distribute across scale and/or parallelise over pixel. This is illustrated in Fig. 13. Computation and distribution of the convolution kernels is not shown explicitly in Fig. 13. These processes form part of the gridding/de-gridding step (denoted ‘GRID’ in the figure). As shown in Fig. 1, these kernels can either be calculated on each compute node per Major Cycle, or calculated once and cached depending on available memory. The calculation of these kernels does not need to be done on the same compute node as the gridding/de-gridding process unless the compute node is memory bandwidth limited.

For the $w$-snapshots implementation of gridding/de-gridding, the FFT step (denoted ‘FFT’ in Fig. 13) also includes a re-projection process for each timestep and the image plane data for each timestep are accumulated at ‘Gather Point A’ in each Major Cycle.

A detailed data distribution diagram for the MS-MFS Major Cycle is shown in Fig. 13. This distribution follows the form of the algorithm as described in [Rau & Cornwell 2011]. The diagram illustrates all potential distribution within the algorithm; however, some aspects of this distribution are more likely to be useful than others. We denote this as ‘CASA-style’ distribution, anticipating the new CASA parallelisation framework.

Further distribution than that shown in Fig. 13 may be useful/necessary. This is due to the manner in which the Taylor term images are constructed. Taylor term images are created using weighted sums of frequency channel images. Consequently, if creation of the frequency channel images has been distributed across $N_{\text{freq}}$ compute nodes then copying all of these images, each with size $N_{\text{pix}}$, to a single compute node to form a Taylor term image may create a memory bandwidth bottleneck. This may be avoided by reducing both $N_{\text{freq}}$ and $N_{\text{pix}}$. For full FOV SKA1 images $N_{\text{pix}} \gg N_{\text{freq}}$ and so it is more beneficial to reduce the size of $N_{\text{pix}}$. This may be done by dividing the full FOV into multiple facets. If this approach is to be taken, distribution by facet should occur at ‘Point A’, as indicated in Fig. 13. Distribution by facet requires an additional phase rotation operation on the continuous visibility dataset prior to the Gridding and FFT steps.

An additional degree of distribution may be introduced at ‘Point A’, by distributing over Stokes parameter. If full-Stokes $A$-projection is being employed, as is expected, then this does not reduce the volume of continuous visibility data in any way but applies differing weighting to the elements of Mueller matrix when forming the convolution kernel for the gridding/de-gridding step. All subsequent steps are then independent between the Stokes parameters.

We note that, if distribution over facets is implemented, the Minor Cycle should update a Global Model, i.e. all facets update the same model. This may be done in a staggered fashion: individual facet models are updated initially, followed by a reconciliation of overlapping facets to remove/amalgamate coincident model components.
Figure 13: MS-MFS Data Distribution Diagram (CASA-style implementation).
If a $w$-snapshots implementation of $w$-projection is being employed then there is an additional gather step, combined with a re-projection step, required within the Major Cycle. This will occur at ‘Point B’, as indicated in Fig. 13.

The ASKAPsoft package distributes the Major Cycle somewhat differently, see Fig. 14. ASKAPsoft uses “all reduce” across all participating ranks corresponding to an individual frequency channel to sum the result of individual degridding processes (for each Taylor term) and broadcast it back, then each rank does its own subtraction from the copy of the measured visibilities, which is then gridded. This communication happens at every timestamp so that all baselines are presented in one accessor structure. Final images are obtained after summing across all frequency channels. Apart from this cross-talk after the de-gridding stage, each frequency and Taylor term is processed independently until the final sum to produce the Taylor terms images that go into the multi-scale convolution step. This ASKAP-style distribution has the disadvantage of duplicating operations by a factor of $N_t$ in the de-gridding step, but the application of Taylor term weights directly to the visibilities in the gridding step, rather than summing weighted images, is beneficial in the case that $N_{\text{pix}}^2 \gg N_{\text{vis}}$.

A proposed distribution scheme for an SKA1 MS-MFS Major Cycle is shown in Fig. 15. This distribution is more similar in nature to the CASA-style distribution illustrated in Fig. 13 and distributes data to compute islands by partitioning in frequency. The reasons for this are (1) to reduce the total number of operations in the forward step; (2) distributing in frequency before distributing in time may be preferable for implementation reasons, which makes (3) gathering in time within each compute island, before gathering in frequency inter-island, prior to the Minor Cycle to minimise the inter-island reduction steps. This third point may be surpassed by more sophisticated implementations which grid directly to Taylor term images; however, further investigation is required to establish the trade-off between reduced total operation counts in this scenario against the memory and memory bandwidth requirements of the later reduc-

![Figure 14: MS-MFS Data Distribution Diagram (ASKAP-style implementation).](image-url)
tion step(s). Specifically, for an \( N_t \) Taylor terms, the image definition for direct gridding needs to extend to \( 2N_t - 1 \) terms. This is required for the computation of cross-terms between Taylor PSFs and is used only in the first Major Cycle. The cross terms \( w_{\text{psf},i} \cdot w_{\text{psf},j} \) need to be computed as \( w_{\text{psf},0} \cdot w_{\text{psf},i+j} \) for FFT based convolutions to be correct, because when visibilities from different frequencies are averaged onto the same grid cell, \( F(w_{\text{psf},i} \cdot w_{\text{psf},j}) \neq F(w_{\text{psf},i}) \cdot F(w_{\text{psf},j}) \), as gridding and frequency-dependent weighting do not commute.

**Compressed Sensing**  Compressive sensing na"ively offers the same type of parallelisation as CLEAN, i.e. the measurement operators could be parallelised on many-core systems. However, since the algorithms are inherently iterative (cf. the Major Cycles of CLEAN) they do not naturally lend themselves to parallelisation across multi-node systems. However, recent developments in proximal splitting algorithms, combined with a further data partitioning [RD26] lead to algorithmic structures that are highly parallelisable across multi-node systems. Proximal splitting algorithms are generally iterative and do not naturally adhere to a structure that can be highly parallelised. However, algorithms such as the parallel proximal algorithm (PPXA) and the simultaneous-direction method of multipliers (SDMM) do offer a parallel implementation structure, where all the proximity operators can be computed in parallel rather than sequentially.

These new algorithmic structures lead to high levels of parallelisation across multi-node systems [RD26]. Furthermore, the structure of these algorithms will not only allow computations to be distributed, but memory and storage requirements also. In the case of radio interferometric data with \( N_{\text{vis}} \) visibilities, these could be partitioned into \( R \) blocks of data with \( R \) measurement operators. In practice, the visibility data would be distributed across \( R \) compute nodes which would then perform the application of the measurement operator in parallel. It is worth noting that the communication overheads at each iteration of this Major Cycle would require the exchange of information regarding the image only and not the visibility data. Such an approach will be critical to tackling the big data-sets anticipated from the SKA. These new highly distributed algorithms are being implemented in the PURIFY package but are not yet ready for use. During the evaluation of of these operators only the model image requires an update per Major Cycle. The continuous visibility data only need to be read once. Issues resulting from holding the continuous visibilities in memory can be mitigated by splitting the full visibility dataset over multiple compute nodes. A distributed compressed sensing Major Cycle is shown in Fig. 16.

### 3.7 Reprojection

#### 3.7.1 Description

Subtraction of an arbitrary plane \( w = au + bv \) in the \((u,v,w)\) space from the baseline coordinates (i.e. reducing \( w \) for each visibility to the amount which depends on associated \( u \) and \( v \)) leads to the following image distortion

\[
\begin{align*}
    l' &= l + a \left( \sqrt{1 - l^2 - m^2} - 1 \right) \\
    m' &= m + b \left( \sqrt{1 - l^2 - m^2} - 1 \right)
\end{align*}
\]

where \((l,m)\) are the true offsets from the tangent point and \((l',m')\) are the offsets seen in the distorted image. The image reprojection step employs 2D interpolation to obtain values for each pixel in the target \((l,m)\) grid from the values in the \((l',m')\) grid.
Figure 15: Proposed MS-MFS Major Cycle Data Distribution indicating interconnects.

Parameters $a$ and $b$ of the $w$-plane can either be obtained by fitting a plane into $u$, $v$ and $w$. 
or computed for a specific time with

\[ a = \tan Z \sin \chi \]
\[ b = -\tan Z \cos \chi, \tag{9} \]

where \( Z \) and \( \chi \) are the local zenith and parallactic angles of the tangent point, respectively.

### 3.7.2 Cost

The cost of re-projection scales with the number of pixels in the image being re-projected, with components for both the co-ordinate generation and the re-projection itself, such that

\[ C_{\text{reproj}} = C_{\text{CG}} + 0(N_{\text{pix}}^2), \]
where
\[ C_{CG} = 0((N_{pix}/N_{dec})^2). \]  

Often this complexity is expressed as a single scaling with \( N_{pix} \) with a pre-factor appropriate to the type of re-projection being employed. ASKAPsoft uses image reprojection code based on casacore, which supports either bilinear interpolation which scales as
\[ C_{reproj, bl} = O(7N_{pix}^2), \]
or bicubic interpolation, which scales as
\[ C_{reproj, bc} = O(48N_{pix}^2). \]

The cost of generating a single realisation of interpolation weights can be quite high (approximately 100 multiply-adds and evaluations of Eq. 8 for the bicubic method). Therefore, as an approximation, the grid can be decimated with a certain factor, \( N_{dec} \), and interpolation matrix can be computed for each group of close pixels rather than for each individual pixel. Additional research is required to explore further optimisation options.

Note, the same reprojection is required for dirty image, weight image and PSF. Without any further approximations, this implies the need for additional (real) pool memory with the same dimensions as the computed image for each reprojected grid.

### 3.7.3 Scalability

Distribution of the re-projection process may be feasible by partitioning the image. However, this would require treatment of over-lapping regions to be handled carefully.

### 3.8 Mosaicking

#### 3.8.1 Description

In the case where a science field requires multiple pointings, a linear mosaic image can be formed using a least squares approach:
\[ \rho_{LM} = \frac{\sum_{pt} w_{pt} A_{pt} I_{pt}^M}{\sum_{pt} w_{pt} A_{pt}^2}. \]

where \( w_{pt} \) is an arbitrary weighting factor of individual pointings (mosaicking weight). Individual pointings here represent both physical pointings of the telescope observed at different times and different beams on the sky for multi-beam instruments observed in parallel. Note, although the weighting factor \( w_{pt} \) could, in principle, be used for optimal weighting of different pointings due to changes in sensitivity, this type of weighting should really be done during gridding and, therefore, included in \( A_{pt} I_{pt} \) and \( A_{pt}^2 \). This is because the sensitivity varies with time and frequency. ASKAPsoft grids the product of visibility and corresponding weight and accumulates the sum of these weights for each convolution function used to form the weights image (weights image can be presented in the form of sensitivity image if scaled appropriately, although only relative weights make physical sense). Additional mosaicking weights \( w_{pt} \) may still be required for multi-beam systems (e.g. SKA1-SURVEY) to account for noise correlation between adjacent electronically formed beams.

In the over-simplified scenario, where the primary beams \( A \) are identical for all antennas, this corresponds to a weighted sum of the dirty images formed by a Fourier sum of the observed visibilities.
The PSF will vary across the linear mosaic, making deconvolution employing a Minor Cycle unsuitable for treating the mosaic as a single field. Rau [REF] notes that deconvolution can be done on a modified version of the linear mosaic by employing an approximate PSF during the Minor Cycle, however we do not pursue this further here as it is more likely to be advantageous to distribute deconvolution over multiple SKA1 scale fields rather than combine fields together. A disadvantage of this is the effect of bright sources outside the field of view on the continuous visibilities.

Note that $A$-projection (and $AW$-Projection) produce numerator and denominator of the mosaicking equation directly. Therefore, calculation of the linear mosaic involves simple summation (with weight $w_{pt}$) of individual images as well as weight images across all fields ($2N_{pix}^2$ multiplies per mosaic field and $N_{pix}^2$ adds per field). For the pure W-Projection algorithm (which does not take the primary beam into account) one has to follow the linear mosaicking equation to compute the result ($4N_{pix}^2$ multiplies per mosaic field and $N_{pix}^2$ adds per field).

Individual pointings of the mosaic (or beams for SKA1-SURVEY and SKA1-LOW) are likely to have separate phase and delay tracking centres (to reduce the $w$-term). Therefore, an image plane reprojection to a common coordinate system would be required before summation. This step, however, can be combined with image plane reprojection used in the $w$-snapshots algorithm, provided the tangent point of the desired mosaic is known at the time of observations.

- We recommend adding a weights/sensitivity image to the list of data products to be able to stitch individual fields at a later stage, if necessary.

### 3.8.2 Cost

The cost of mosaicking is

$$C_{mos} = C_{re-proj} + 0(N_{pix}^2),$$

with approximately $5N_{pix}^2$ operations per individual mosaic field for the summation. The cost of re-projection can be discarded if this is done for each field as part of the $w$-snapshots re-projection.

### 3.8.3 Scalability

In the mosaicking process, each pixel is processed independently. Consequently the process can be parallelised by image region.

### 3.9 De-dispersion

#### 3.9.1 Description

The intervening interstellar and inter-galactic medium causes two effects relevant to the detection of pulsed or transient emission: (1) a frequency dependent delay term causing dispersion of the signal in time as a function of frequency; (2) scattering of the signal causing a broadening of the intrinsic pulse width. The first of these can be corrected for by staggered summation of individual frequency channels; the second cannot be corrected.

Correcting the effect of dispersion leads to an increase in signal to noise proportional to $\sqrt{N_{chan}}$ (neglecting the effects of spectral behaviour) and so searching for transient sources in de-dispersed data can be beneficial for recovering low signal-to-noise objects. In Table 4 we summarise a number of important characteristics of this plot. $DM_{max}$ is the $DM$ at the
Figure 17: Parameter space for the detection of pulsed transient radio sources: pulse width against dispersion measure. For each SKA band, the dotted line shows the dispersive delay between its maximum and minimum frequencies; dedispersion is unnecessary for pulses with a width exceeding this. Dashed lines show the scattering timescale from the model of [RD28]; pulses shorter than this will be scatter-broadened out to this timescale. (Note, however, that this is an approximate, empirical relationship, with source-to-source variation of up to two orders of magnitude.) For each SKA band, the shaded region shows the region of the parameter space in which dedispersion is beneficial. Also shown are the pulse widths and dispersion measures of some known transient objects: small black dots show the population of known pulsars [RD29], and large red dots and limits show published fast radio bursts (FRBs) [RD30]. Figure credit: Justin Bray.

In practice, it is never worth dedispersing with spacing of $\Delta DM$ up to a $DM$ of $DM_{\text{max}}$. $DM_{\text{max}}$ only applies for long pulses, much longer than the dump time, so only a very coarse spacing is needed. (At $DM_{\text{max}}$ the number of required $DM$ trials is of order unity.) The maximum number of required $DM$ trials, $N_{DM,\text{max}}$, is actually for a pulse with an inherent width equal to the dump time and is given by the ratio between $\Delta DM$ and $DM_{\text{worst}}$, which is the $DM$ at which a pulse is scatter-broadened out to the same timescale as the dump time. Table 4 lists $N_{DM,\text{max}}$ for a number of different SKA1 bands.
Table 4: Summary of De-dispersion characteristics.

<table>
<thead>
<tr>
<th>BAND</th>
<th>$DM_{\text{max}}$ (pc cm$^{-3}$)</th>
<th>$\Delta DM$ (pc cm$^{-3}$)</th>
<th>$DM_{\text{worst}}$ (pc cm$^{-3}$)</th>
<th>$N_{DM,\text{max}}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>LOW</td>
<td>667.7</td>
<td>0.4</td>
<td>174.1</td>
<td>435</td>
</tr>
<tr>
<td>MID1</td>
<td>1003.1</td>
<td>2.7</td>
<td>380.9</td>
<td>141</td>
</tr>
<tr>
<td>MID2</td>
<td>1135.6</td>
<td>24.6</td>
<td>624.8</td>
<td>25</td>
</tr>
<tr>
<td>MID3</td>
<td>1355.2</td>
<td>74.2</td>
<td>876.0</td>
<td>12</td>
</tr>
<tr>
<td>MID4</td>
<td>1600.3</td>
<td>213.5</td>
<td>1194.1</td>
<td>6</td>
</tr>
<tr>
<td>MID5</td>
<td>2243.6</td>
<td>458.9</td>
<td>1801.3</td>
<td>4</td>
</tr>
<tr>
<td>SUR1</td>
<td>942.9</td>
<td>10.4</td>
<td>455.0</td>
<td>46</td>
</tr>
<tr>
<td>SUR2</td>
<td>1181.0</td>
<td>35.4</td>
<td>687.3</td>
<td>19</td>
</tr>
<tr>
<td>SUR3</td>
<td>1531.8</td>
<td>189.3</td>
<td>1128.1</td>
<td>6</td>
</tr>
</tbody>
</table>

3.9.2 Cost

The de-dispersion process is essentially a staggered sum of snapshots at different frequencies. Consequently, for an image plane based summation, it scales as

$$ C_{\text{dedisp}} = 0 \left( \sum_{i=1}^{N_{DM,\text{max}}} N_{\text{freq},i}N_{\text{pix}}^2 \right). $$

This sum could alternatively be done at the $uv$-grid stage, or on continuous visibility data.

3.9.3 Scalability

Due to the pairwise nature of the de-dispersion process it can be highly parallel; however, it also has a high data transfer overhead. In a Slow Transients Pipeline distributed over frequency the de-dispersion step could be implemented after $uv$-gridding. In this scenario, one would then gather only the non-zero elements of the distributed $uv$-grids to minimise the interconnect data rates. From these reconstructed grids, a suitable number of $DM$ trials could be constructed in $uv$-space before being Fourier transformed (either by FFT or SFT) to image space. In terms of operations this would be beneficial as the total number of $DM$ trials, $N_{DM,\text{max}}$, is likely to be significantly smaller than the degree of frequency distribution - with perhaps the exception of SKA1-LOW, see Table 4. For optimal signal-to-noise searches should also be done in pulse length. This requires a tree-like addition of adjacent time steps which likely can be done intra-island before inter-island connections are made for dispersion measure searches. This additional search dimension will increase the number of FFT operations by approximately one order of magnitude.

3.10 Data Product Creation

3.10.1 Description

Many of the required data products will be produced as by-products of the Major Cycle. This section outlines the cost of producing these data-products independently of the Major Cycle process. A potential data distribution for this process is illustrated in Fig. 18.
3.10.2 Cost

The costs of producing the data products are comparable to those of a single Major Cycle. To produce the “clean” maps for each Taylor term model image requires convolution of the output model by the clean beam, which has the complexity:

\[ C_{\text{cmap}} = 0(N_{\text{ch}}N_{\text{pix}}^2N_{\text{cb}}^2), \]

where \( N_{\text{ch}} \) is the support of the clean beam in pixels and \( N_{\text{pix}}^2 \) is the number of pixels in each Taylor term model image. For imaging distributed by facets this will be smaller than the number of pixels required for the full FOV, but will have the additional cost of mosaicking the facets to produce a single image, if required.

To produce the residual Taylor term images a set of model visibilities must be produced and subtracted from the measured visibilities \( (C_{\text{sub}}) \), followed by standard imaging assuming \( w \)-snapshots \( (C_{\text{im}}) \), which can then be used to produce Taylor term residual images using a weighted sum \( (C_{\text{rt}}) \):

\[ C_{\text{res}} = 0(C_{\text{res,vis}} + 2C_{\text{im}} + C_{\text{rt}}), \]

where

\[ C_{\text{res,vis}} = C_{\text{sum}} + C_{\text{FFT}} + C_{\text{grid}} + C_{\text{sub}} \]
\[ = 0(N_{\text{pix}}^2 + N_{\text{pix}}^2 \log_2 N_{\text{pix}}^2 + N_{\text{pix}}^2 N_{\text{kernel}}^2 + N_{\text{vis}}) \]

and

\[ C_{\text{im}} = C_{\text{grid}} + C_{\text{FFT}} + C_{\text{re-proj}} + C_{\text{sum}} \]
\[ = 0(N_{\text{vis}}^2 N_{\text{kernel}}^2 + N_{\text{pix}}^2 \log_2 N_{\text{pix}}^2 + 48N_{\text{pix}}^2 + N_{\text{pix}}^2) \]

and

\[ C_{\text{rt}} = C_{\text{sum}} \]
\[ = 0(N_{\text{pix}}^2). \]

The factor of two multiplying the \( C_{\text{im}} \) term in Eq. 11 accounts for the additional processing required to produce the accompanying dirty beam image.

3.10.3 Scalability

The exact value of \( N_{\text{vis}} \) here depends on the degree to which the data have been distributed. We assume as a minimum that distribution in frequency and time have taken place, see Fig. 18, but additional distribution by beam and Stokes parameter would also be likely. In addition, distribution by facet will reduce the value of \( N_{\text{pix}}^2 \). However, in this case an additional (final) mosaicking step would be required to produce a single image for the full FOV.
Figure 18: Data Product Creation Data Distribution Diagram.
“imaging_supporting_document” History

- Document created by Verity Allan (vla22@mrao.cam.ac.uk) on February 09, 2015 - 2:21 PM GMT - IP address: 131.111.185.15

- Document emailed to Ronald Nijboer (rnijboer@astron.nl) for signature on February 09, 2015 - 2:22 PM GMT

- Document viewed by Ronald Nijboer (rnijboer@astron.nl) on February 09, 2015 - 3:03 PM GMT - IP address: 192.87.1.49

- Document e-signed by Ronald Nijboer (rnijboer@astron.nl) on February 09, 2015 - 3:03 PM GMT - Time Source: server - IP address: 192.87.1.49

- Document emailed to Paul Alexander (pa@mrao.cam.ac.uk) for signature on February 09, 2015 - 3:03 PM GMT

- Document viewed by Paul Alexander (pa@mrao.cam.ac.uk) on February 09, 2015 - 6:42 PM GMT - IP address: 131.111.185.15

- Document e-signed by Paul Alexander (pa@mrao.cam.ac.uk) on February 09, 2015 - 6:42 PM GMT - Time Source: server - IP address: 131.111.185.15

- Signed document emailed to Verity Allan (vla22@mrao.cam.ac.uk), Ronald Nijboer (rnijboer@astron.nl) and Paul Alexander (pa@mrao.cam.ac.uk) on February 09, 2015 - 6:42 PM GMT