Data processing with the RTS

A GPU-accelerated calibration & imaging stream processor

Daniel Mitchell
2018 ICRAR/CASS Radio School

The RTS (Real-Time System)

A GPU-accelerated calibration & imaging stream processor designed for MWA

- Design Drivers
- Design Solutions
- Data & Processing Flow
- Using the RTS

Design Drivers

Design Drivers

- Very wide MWA field of view
  - Standard W-projection approach requires very large gridding kernels.
- Variable, highly polarised primary beams
  - Varying across the field of view and in time as a field is tracked.
  - Different tile beams due to analogue beamformer variability.
- Ionospheric refraction
  - Propagation through the ionosphere causes ≈ λ^2-dependent delays (λ-dependent phases). Kolmogorov turbulence + wave-like structures.

    → time-variable phase fluctuations across the aperture and the FoV.
- Challenging to couple very large W terms with highly variable A and I terms.
- Excellent MWA snapshot uv coverage and synthesised beam

Variable Polarised Primary Beams

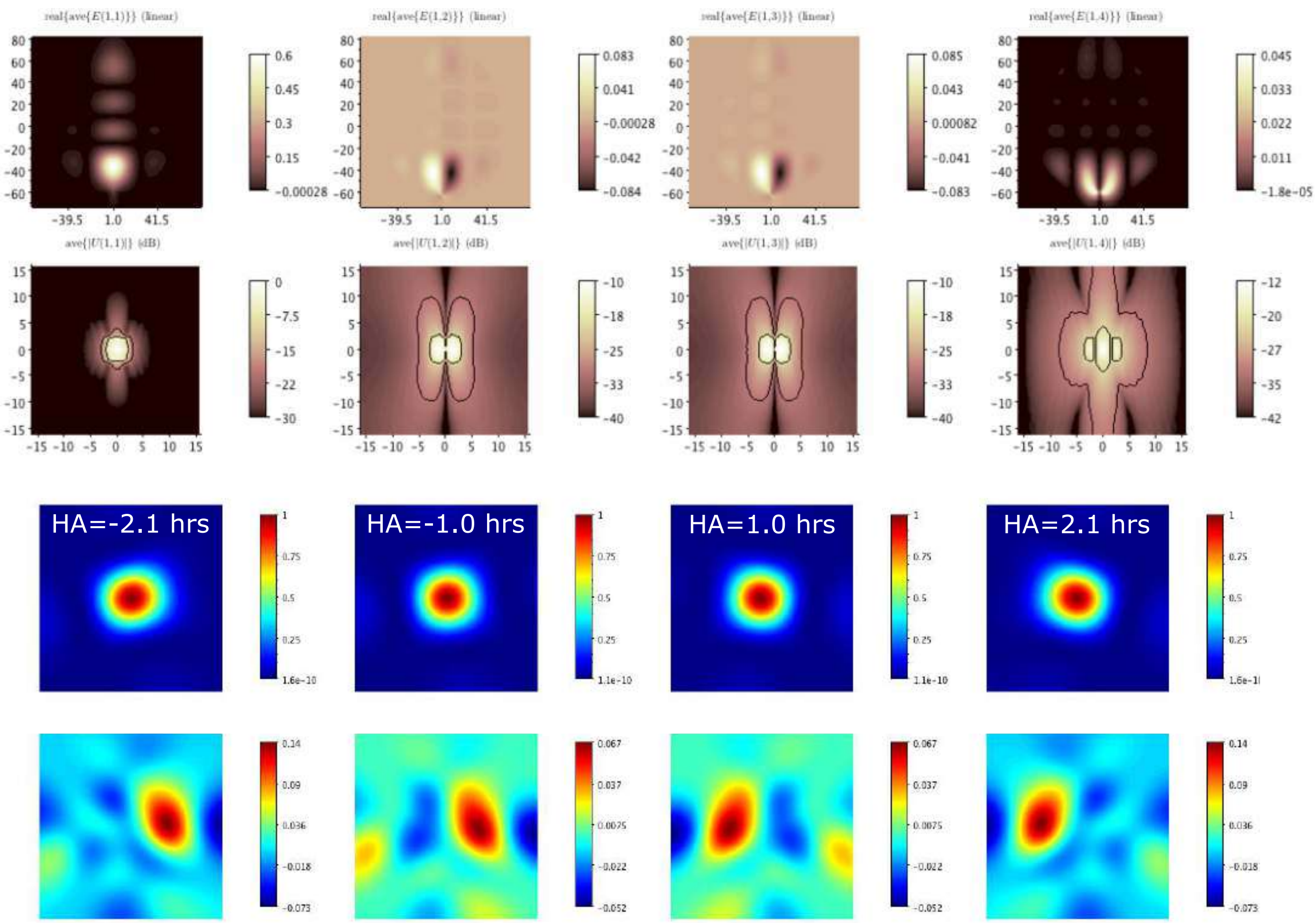
NN←xx NN←xy NN←yx NN←yy

All-sky polarised tile response

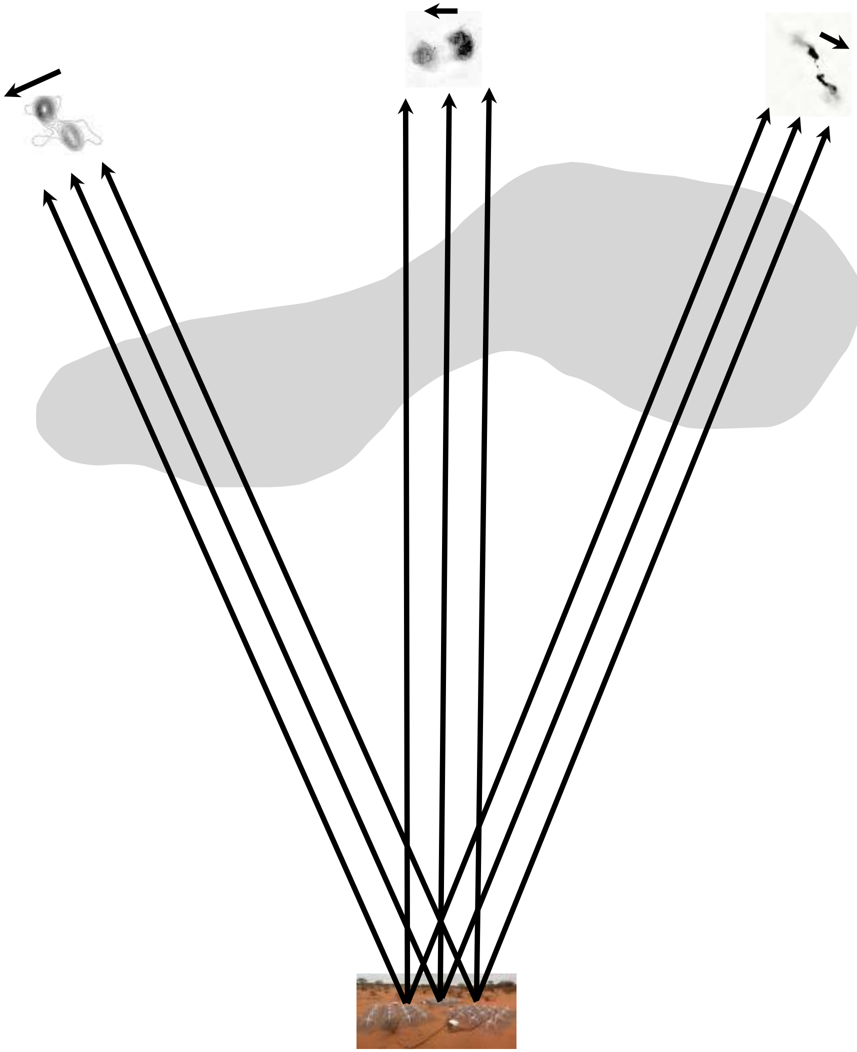
Fourier response

Time-dependent tile response

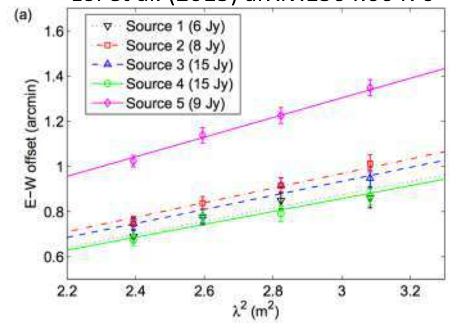
Time-dependent error



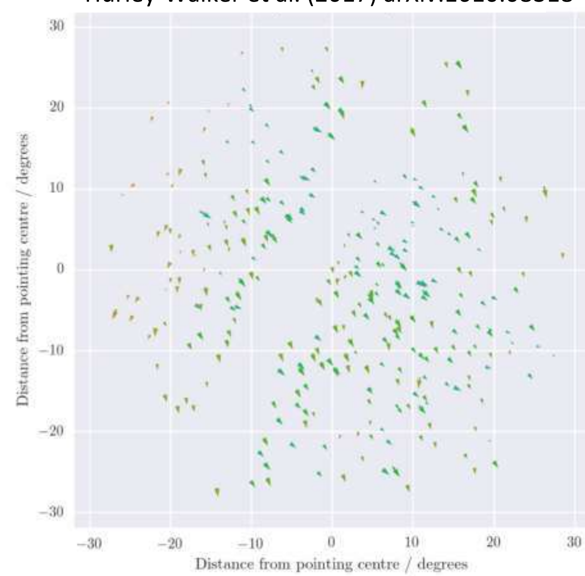
Ionospheric Refraction



Loi et al. (2015) arXiv:1504.06470



Hurley-Walker et al. (2017) arXiv:1610.08318



Design Solutions

Accumulate Residual Snapshot Images

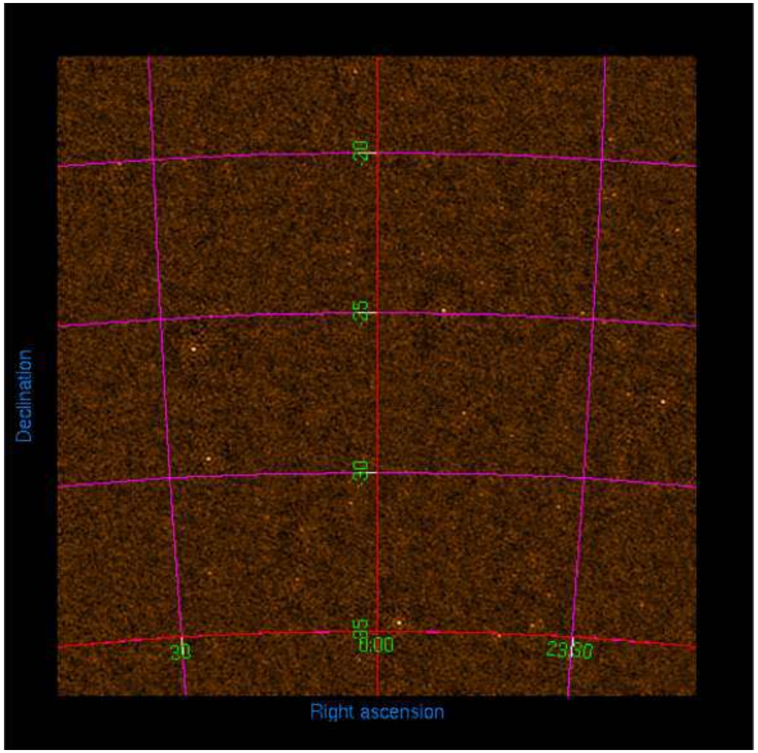
- Take advantage of MWA strengths: it is an ionospheric machine!
  - Lots of relatively short baselines.
  - Solve for the ionosphere rather than direction-dependent tile phases
    – increase SNR and decrease the number of free parameters.
  - Segment visibility data in time during imaging
    ≈ linear ionospheric variations.
    ≈ 2D phases in Fourier transform.
    ≈ equal primary beams for each visibility.

- Well suited to stream processing and real-time operation
  - Parallelize in frequency for high-throughput stream processing.
  - Use a cluster of GPUs to reduce power and cost per FLOP.
  - Extend data reduction averaging times by moving to the image domain.

Peeling

- Subtract initial sky model
- Add strong sources back one-by-one, redo calibration for each and re-subtract

$$E_j = \left(\sum_{k\neq j} V_{jk}\, M_{jk}^{\dagger}\right)\left(\sum_{k\neq j} M_{jk}\, M_{jk}^{\dagger}\right)^{-1}; \quad M_{jk} = J_j\, S\, J_k^{\dagger}\ ; \quad V_{jk} = M_{jk} + N_{jk}$$

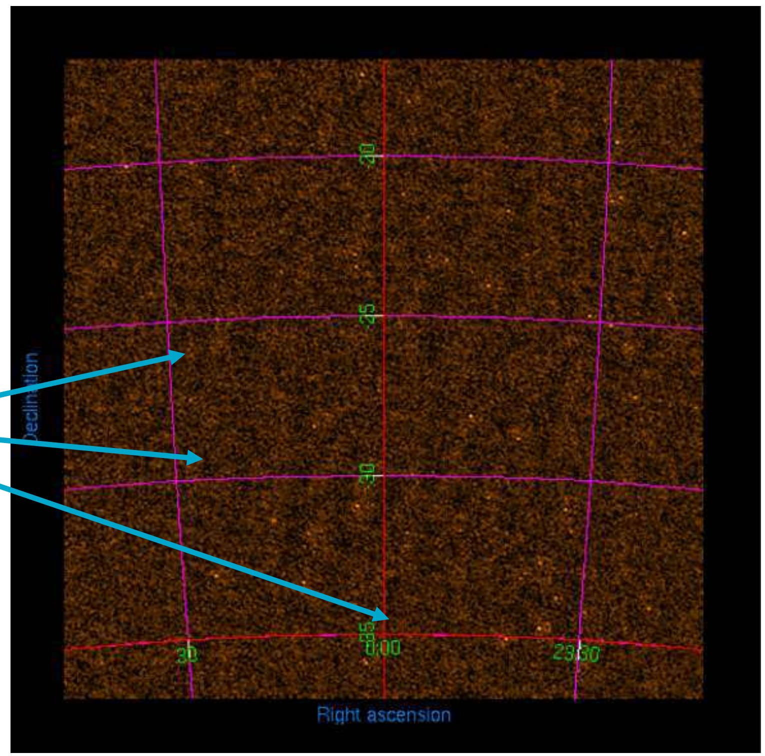


Peeling

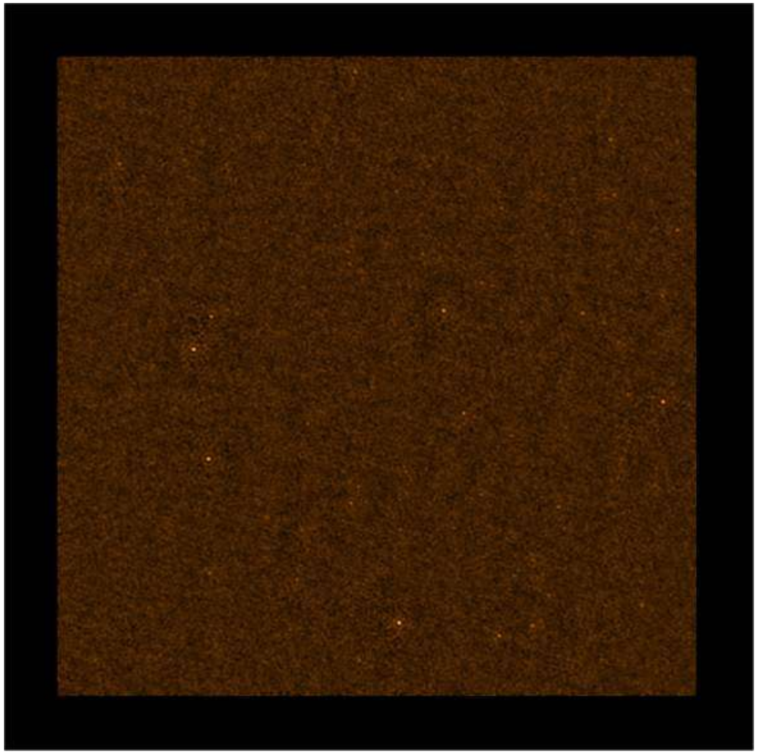
- Subtract initial sky model
- Add strong sources back one-by-one, redo calibration for each and re-subtract

$$E_j = \left(\sum_{k\neq j} V_{jk}\, M_{jk}^{\dagger}\right)\left(\sum_{k\neq j} M_{jk}\, M_{jk}^{\dagger}\right)^{-1}; \quad M_{jk} = J_j\, S\, J_k^{\dagger}\ ; \quad V_{jk} = M_{jk} + N_{jk}$$

Peel the 5 strongest

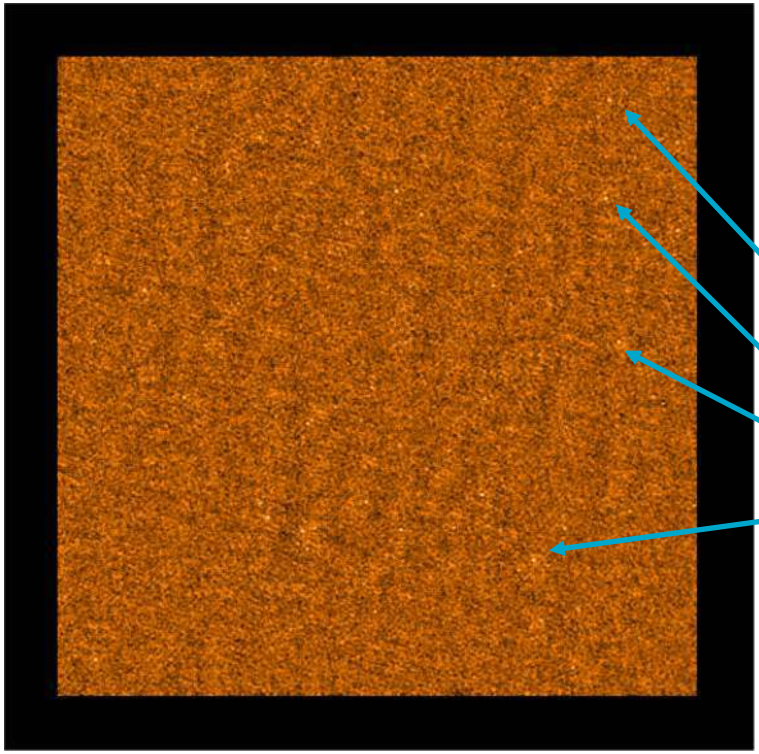


Peeling — over-peeling



Dirty image with direction-independent calibration

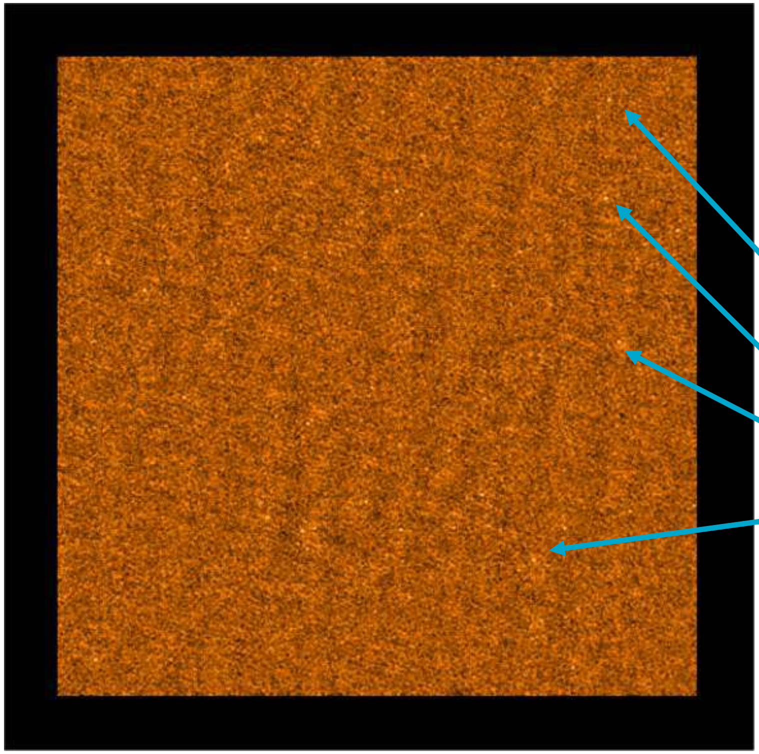
Peeling — over-peeling



Dirty image with direction-independent calibration and 50 sources subtracted (but not peeled)

The residuals are dominated by weaker sources, not by subtraction artefacts.

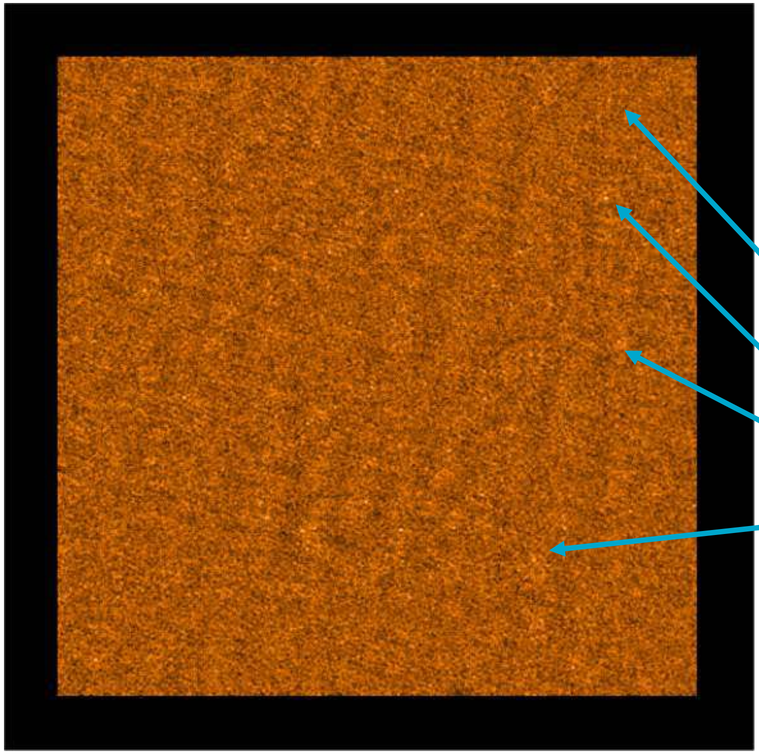
Peeling — over-peeling



Dirty image with direction-independent calibration and 50 sources subtracted (5 of the 50 peeled)

Peeling the 5 brightest sources doesn't have too much of an effect on the residuals.

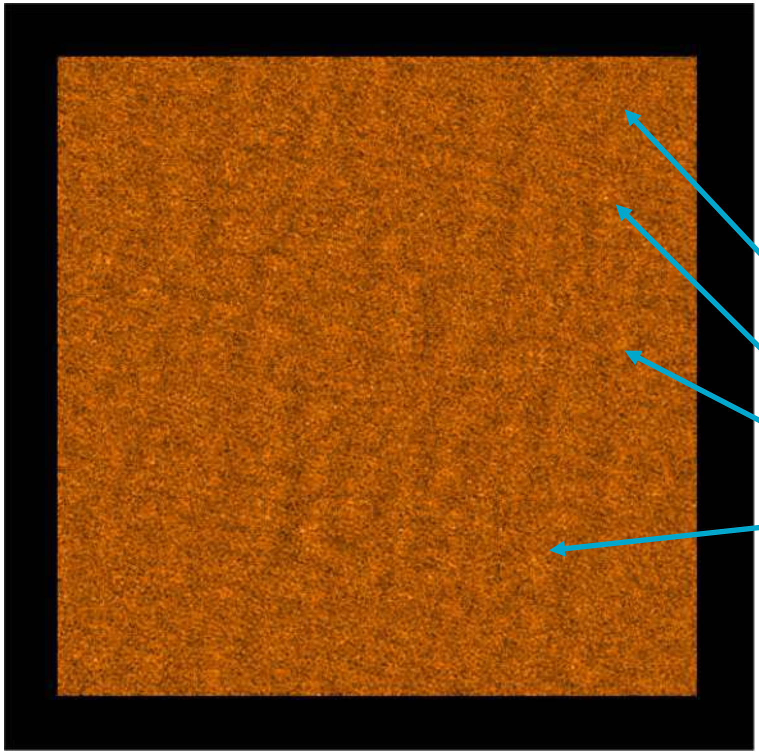
Peeling — over-peeling



Dirty image with direction-independent calibration and 50 sources subtracted (10 of the 50 peeled)

But as more of the 50 sources are peeled the weaker sources and their sidelobes disappear!

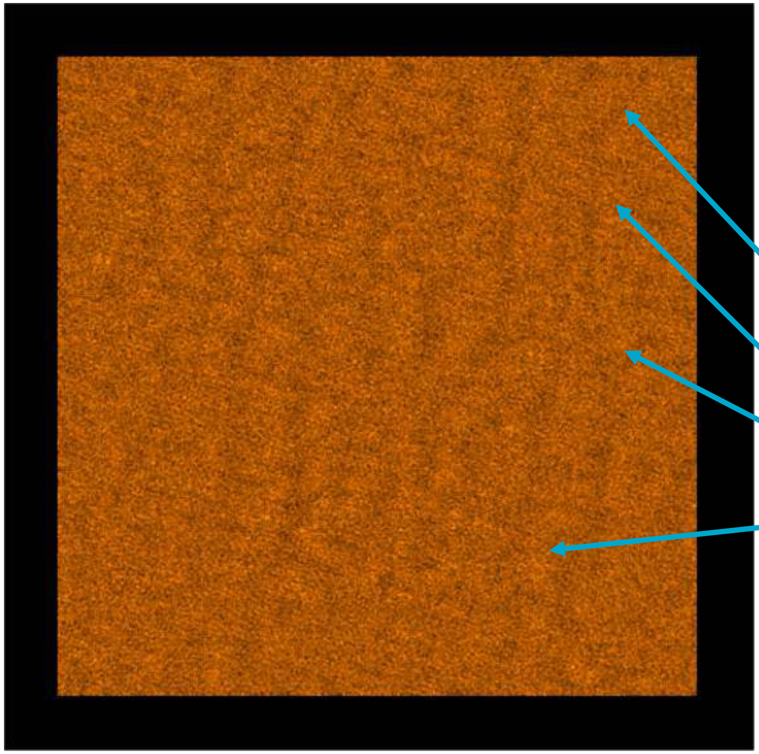
Peeling — over-peeling



Dirty image with direction-independent calibration and 50 sources subtracted (20 of the 50 peeled)

But as more of the 50 sources are peeled the weaker sources and their sidelobes disappear!

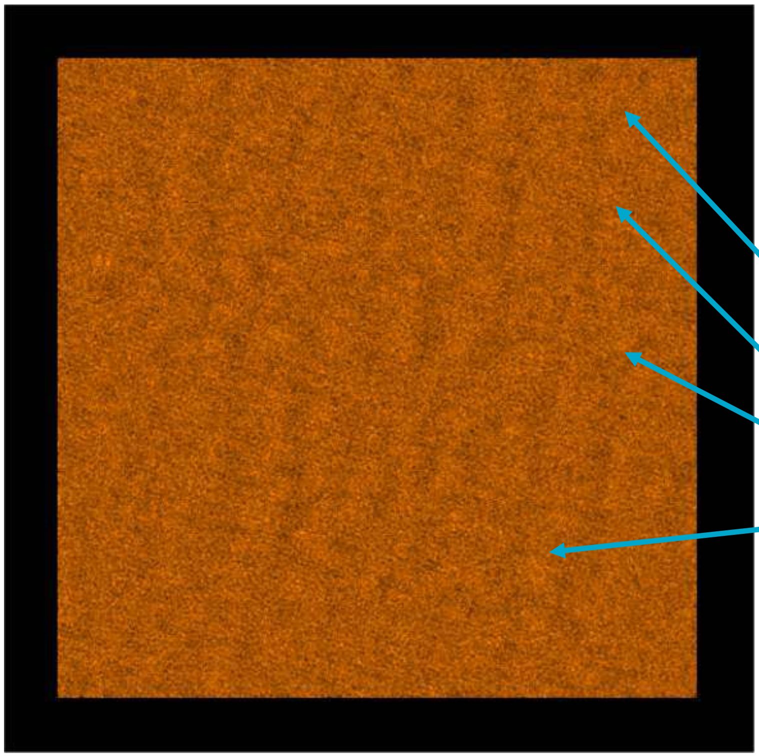
Peeling — over-peeling



Dirty image with direction-independent calibration and 50 sources subtracted (30 of the 50 peeled)

But as more of the 50 sources are peeled the weaker sources and their sidelobes disappear!

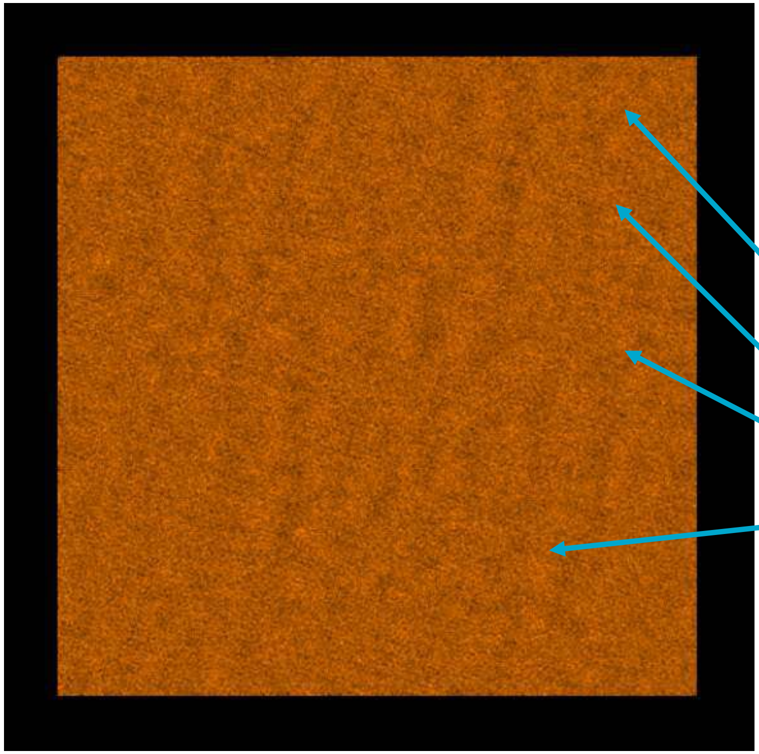
Peeling — over-peeling



Dirty image with direction-independent calibration and 50 sources subtracted (40 of the 50 peeled)

But as more of the 50 sources are peeled the weaker sources and their sidelobes disappear!

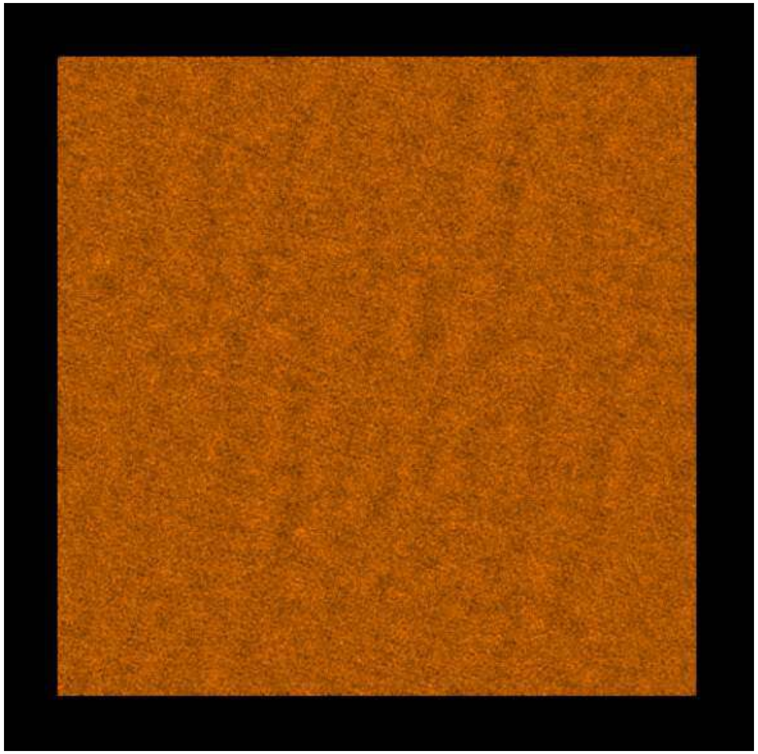
Peeling — over-peeling

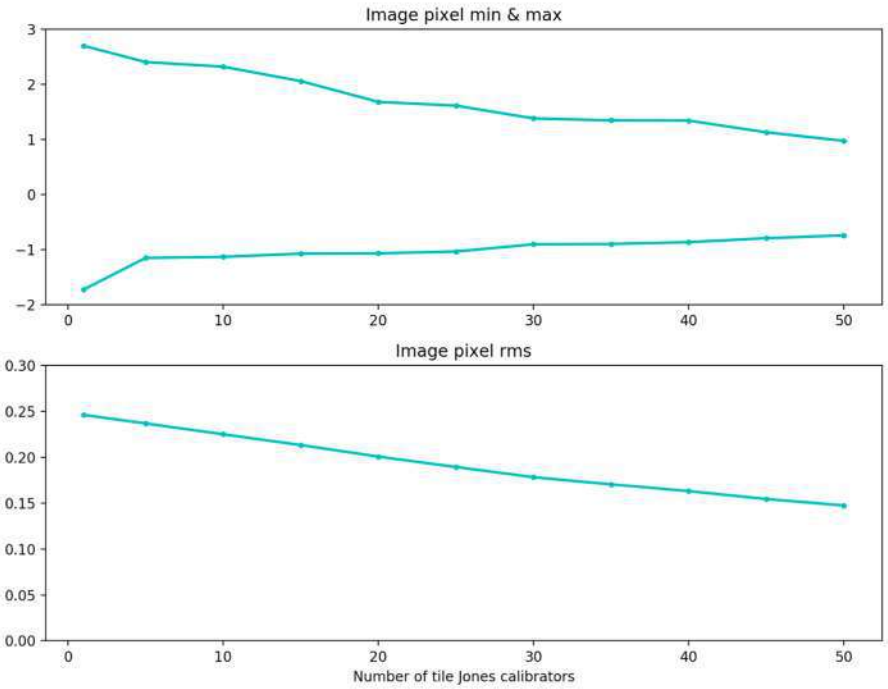


Dirty image with direction-independent calibration and 50 sources subtracted (all 50 peeled)

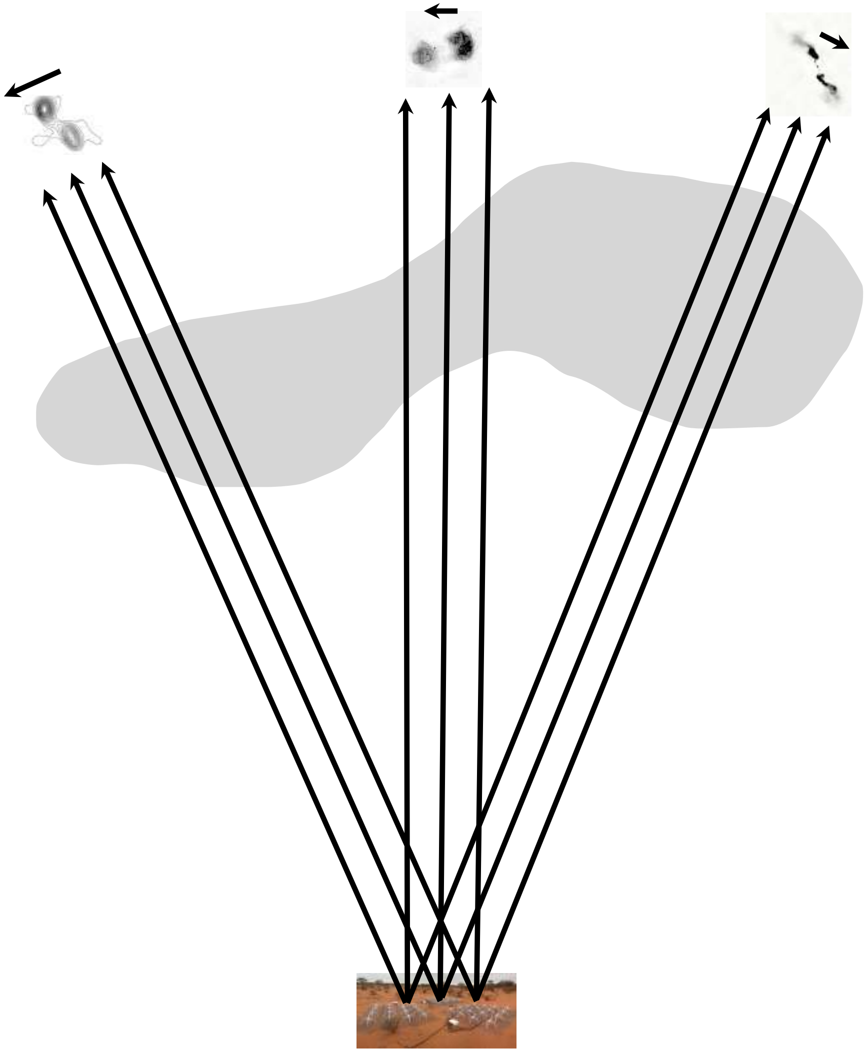
But as more of the 50 sources are peeled the weaker sources and their sidelobes disappear!

Peeling — over-peeling

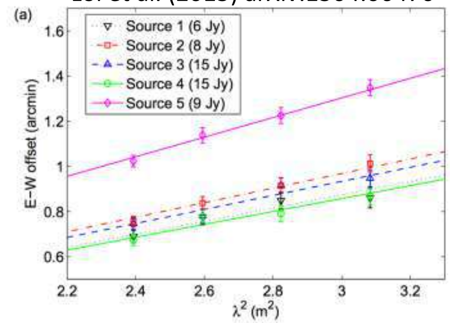




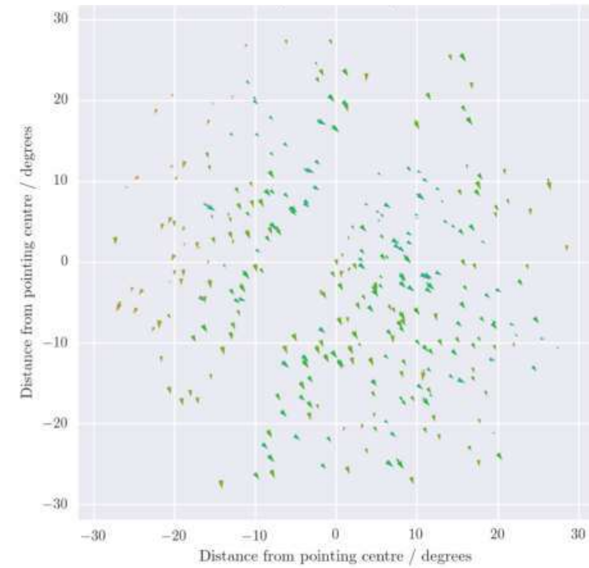
Ionospheric Refraction



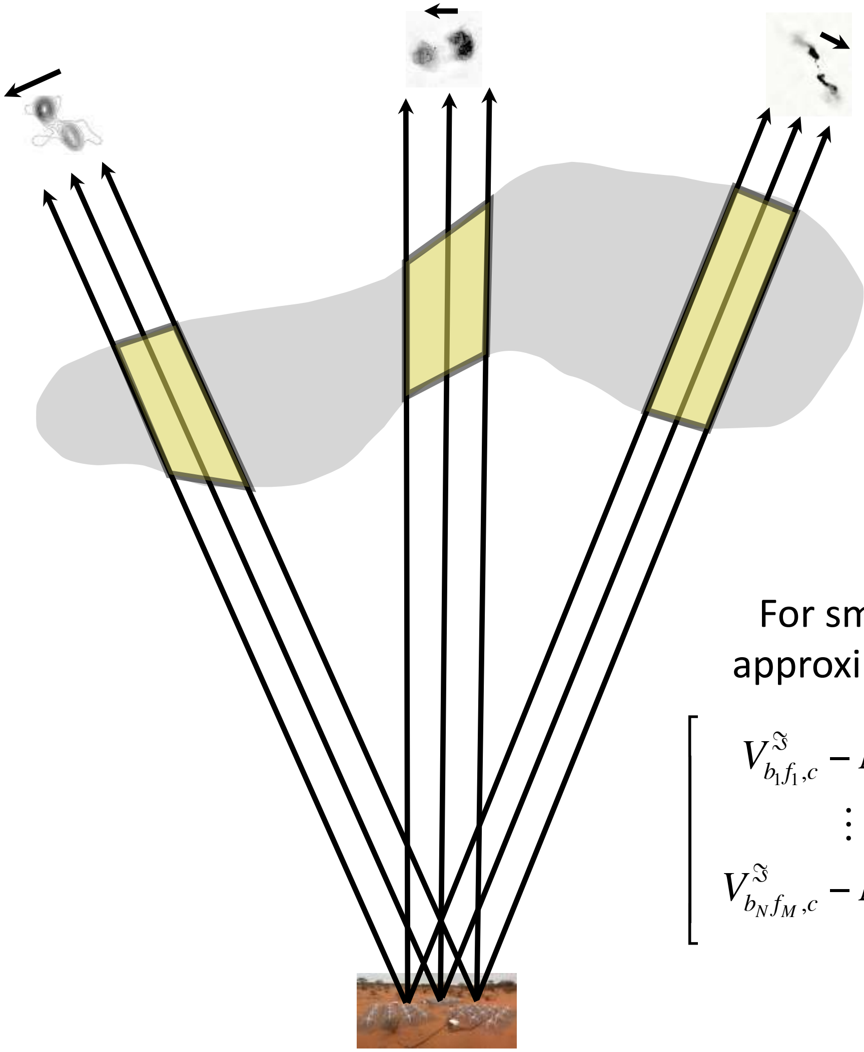
Loi et al. (2015) arXiv:1504.06470



Hurley-Walker et al. (2017) arXiv:1610.08318



Constrained Peeling → linear phase model



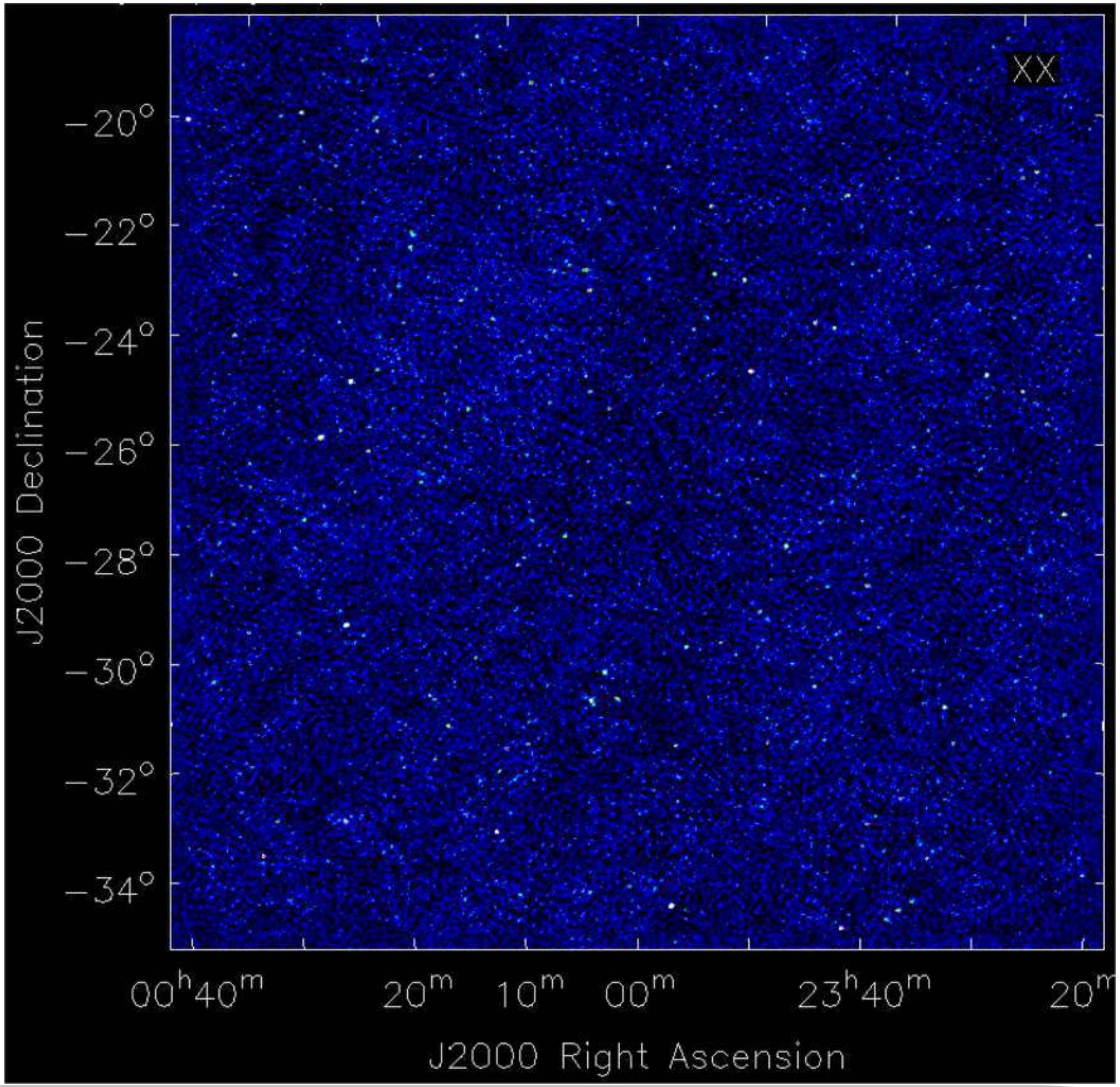
Consider a set of visibilities with:
- a short time duration (≈ 10 sec)
- all-but calibrator *c* subtracted

$$V_{bf,c} \approx N_{bf} + M_{bf,c} \exp\left\{-i2\pi\lambda_f^2\left(u_{bf}\alpha_{l,c} + v_{bf}\alpha_{m,c}\right)\right\}$$

For small offsets ($\lambda^2\alpha$ << synthesised beam size) imag{$V_{bf,c}$} is approximately linear in the α parameters => linear least-squares.

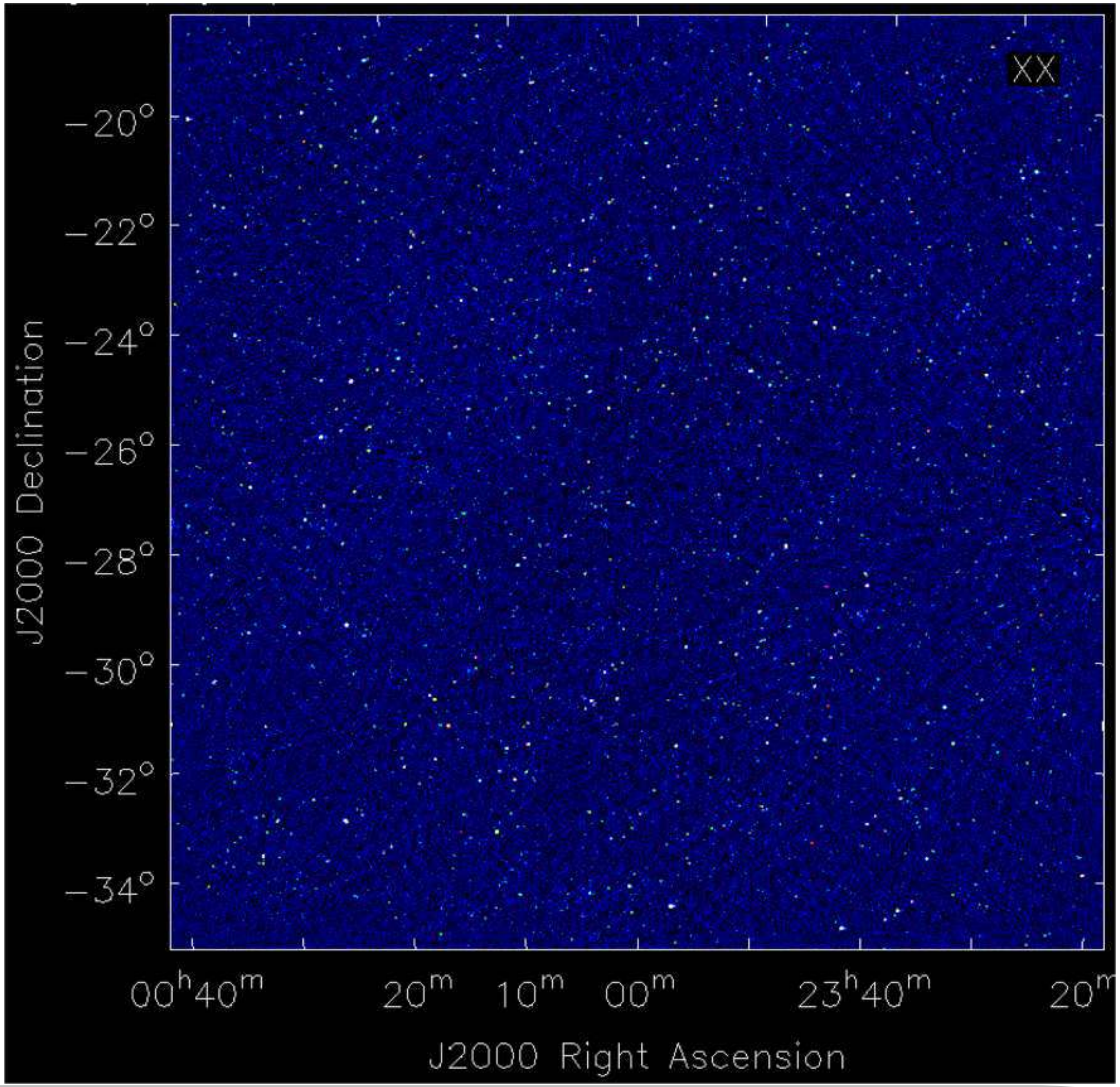
$$\begin{bmatrix} V^{\Im}_{b_1 f_1,c} - M^{\Im}_{b_1 f_1,c} \\ \vdots \\ V^{\Im}_{b_N f_M,c} - M^{\Im}_{b_N f_M,c} \end{bmatrix} \approx -2\pi \begin{bmatrix} \lambda^2_{f_1} M^{\Re}_{b_1 f_1,c} u_{b_1 f_1} & \lambda^2_{f_1} M^{\Re}_{b_1 f_1,c} v_{b_1 f_1} \\ \vdots & \vdots \\ \lambda^2_{f_M} M^{\Re}_{b_N f_M,c} u_{b_N f_M} & \lambda^2_{f_M} M^{\Re}_{b_N f_M,c} v_{b_N f_M} \end{bmatrix} \begin{bmatrix} \alpha_{l,c} \\ \alpha_{m,c} \end{bmatrix}$$

MWA data: 8 seconds × ~ 1 MHz (182 MHz)



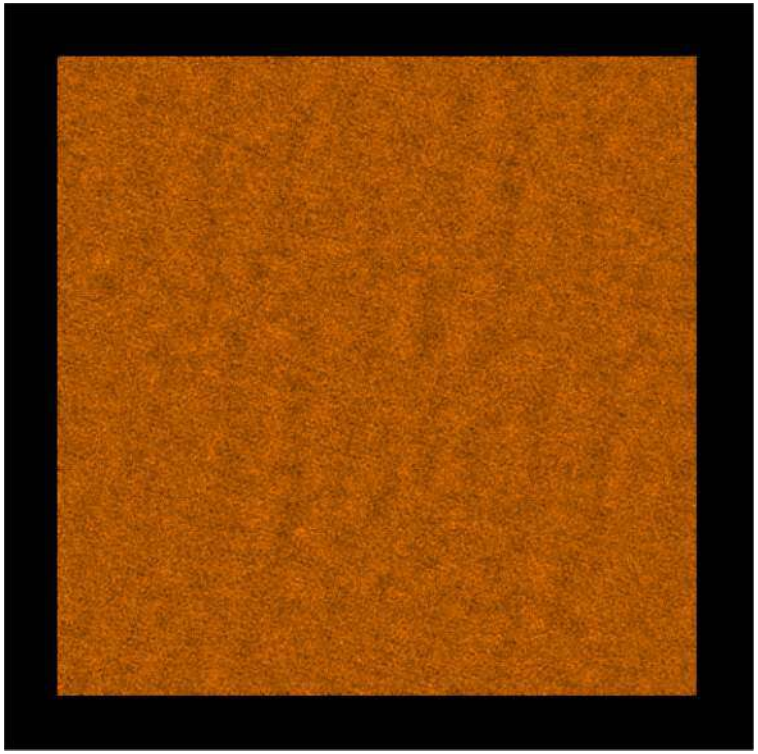
Colour scale ([-1σ - +10σ]):
[-0.19, 1.9] Jy/beam

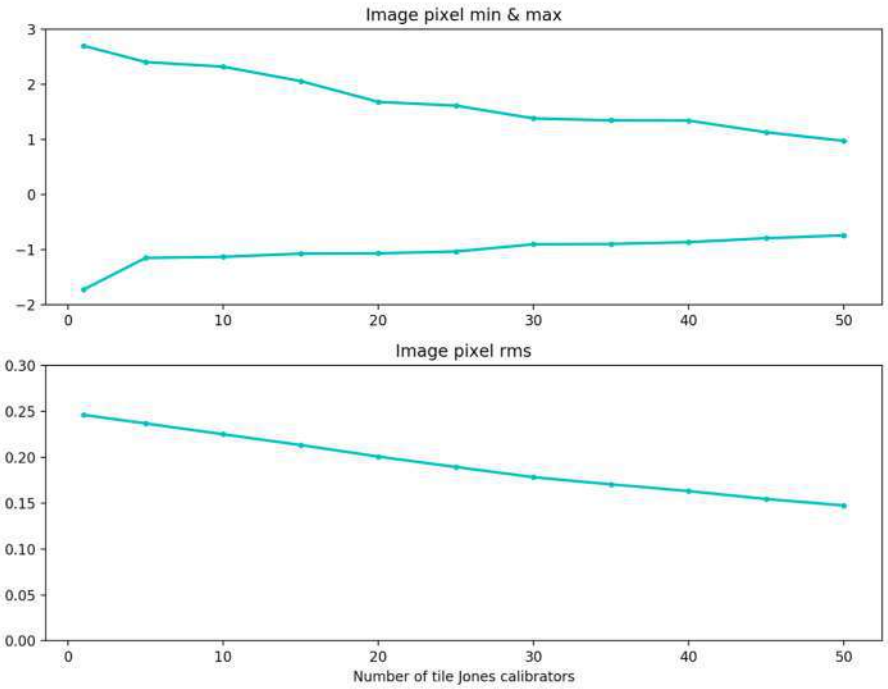
MWA data: 8 seconds × ~ 31 MHz (182 MHz)



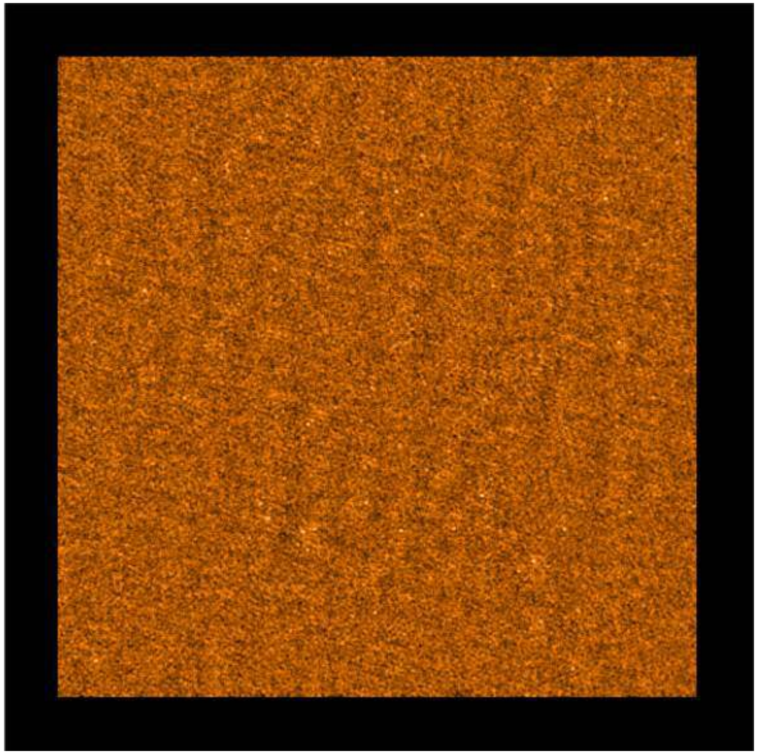
Colour scale ([-1σ - +10σ]):
[-0.06, 0.6] Jy/beam

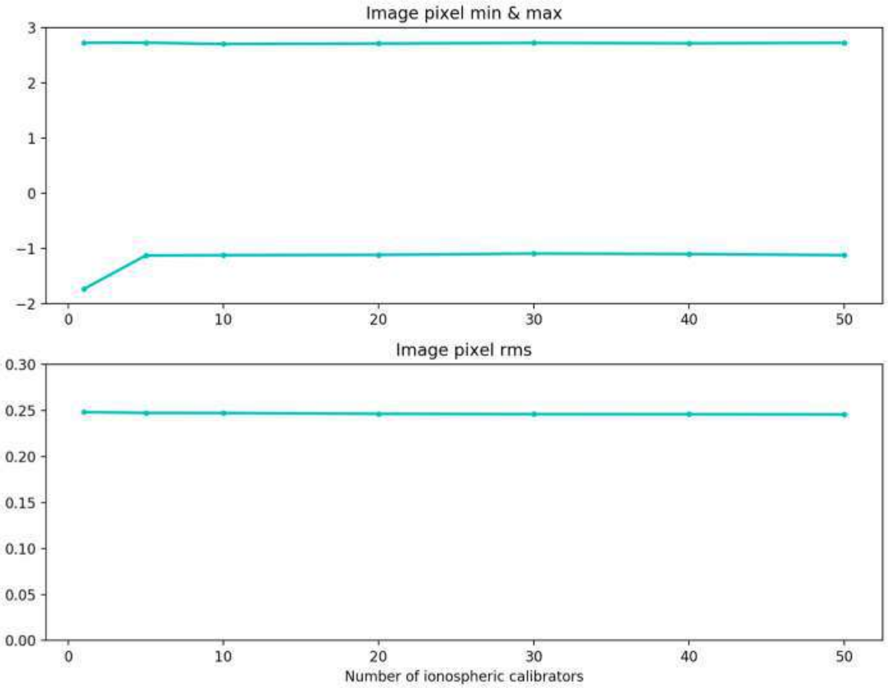
Peeling — over-peeling



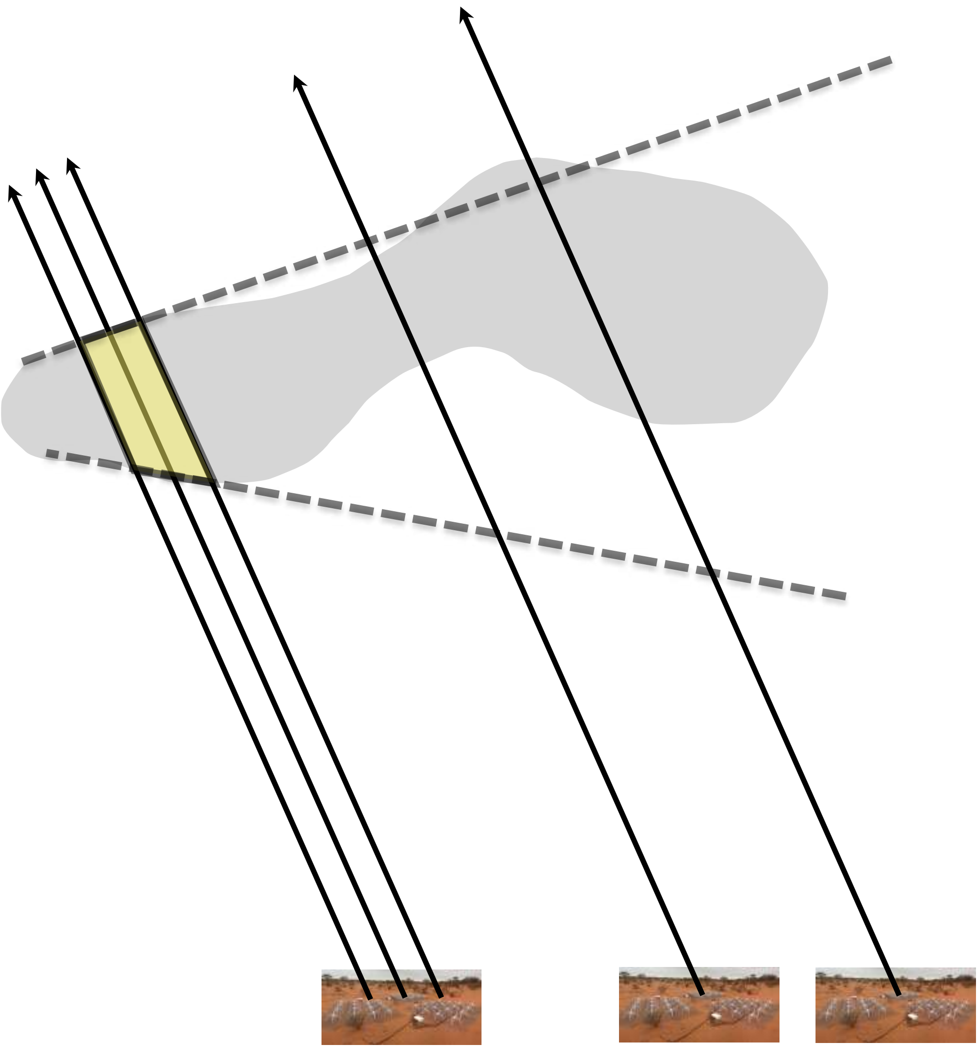


Peeling — over-peeling

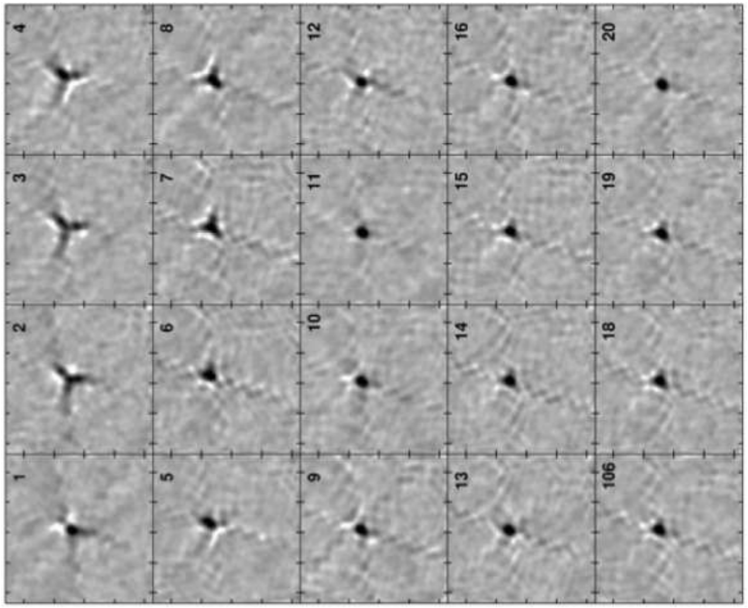


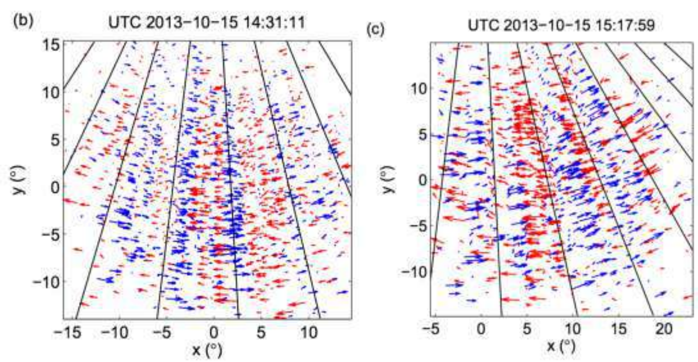


Non-linear ionospheric phases



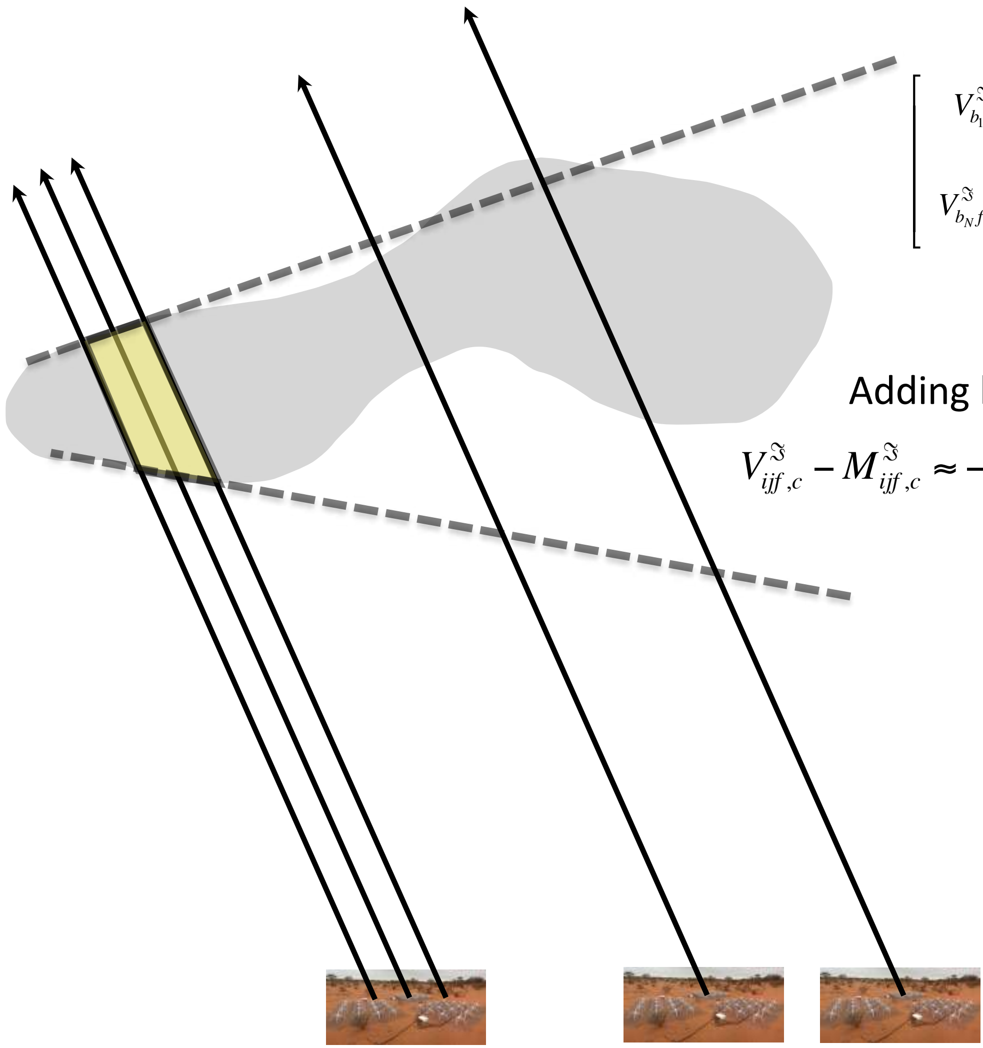
Cotton (2004) ASP Conf. Series 345, 74 MHz, 1-min VLA snapshots





Loi et al. (2015) arXiv:1504.06470

Constrained peeling → higher order models

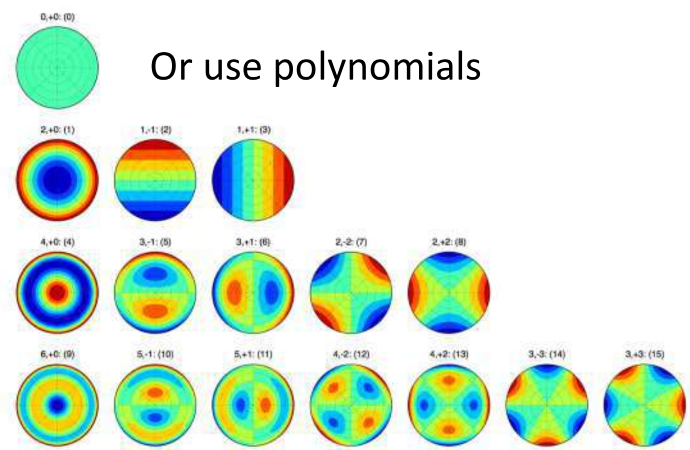


$$\begin{bmatrix} V^{\Im}_{b_1 f_1,c} - M^{\Im}_{b_1 f_1,c} \\ \vdots \\ V^{\Im}_{b_N f_M,c} - M^{\Im}_{b_N f_M,c} \end{bmatrix} \approx -2\pi \begin{bmatrix} \lambda_{f_1}^2 M^{\Re}_{b_1 f_1,c} u_{b_1 f_1} & \lambda_{f_1}^2 M^{\Re}_{b_1 f_1,c} v_{b_1 f_1} \\ \vdots & \vdots \\ \lambda_{f_M}^2 M^{\Re}_{b_N f_M,c} u_{b_N f_M} & \lambda_{f_M}^2 M^{\Re}_{b_N f_M,c} v_{b_N f_M} \end{bmatrix} \begin{bmatrix} \alpha_{l,c} \\ \alpha_{m,c} \end{bmatrix}$$

Adding higher-order phase terms is straightforward:

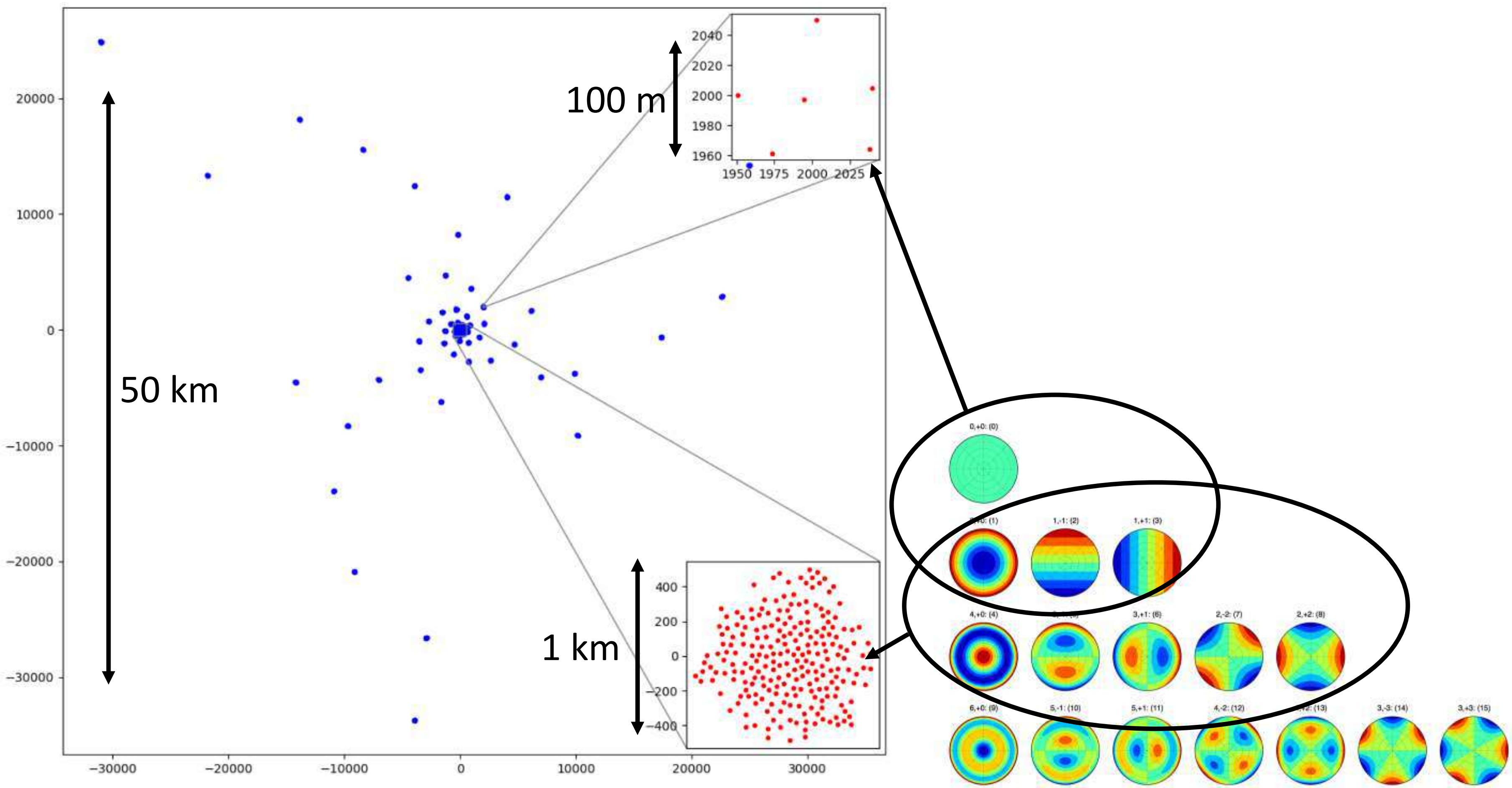
$$V^{\Im}_{ijf,c} - M^{\Im}_{ijf,c} \approx -2\pi\lambda_f \left(dx_{ijf}\alpha_{l,c} + \left(x_{if}^2 - x_{jf}^2 \right)\beta_{l,c} + dy_{ijf}\alpha_{m,c} + \left(y_{if}^2 - y_{jf}^2 \right)\beta_{m,c} \right)$$

Or use polynomials

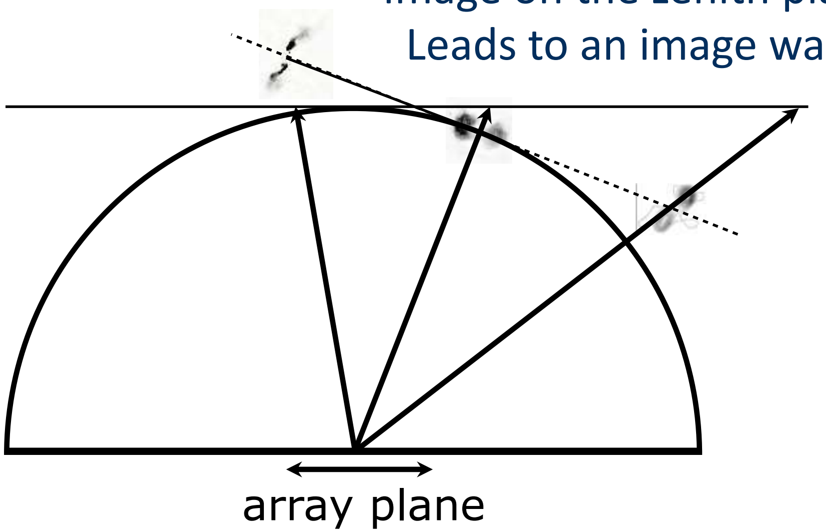


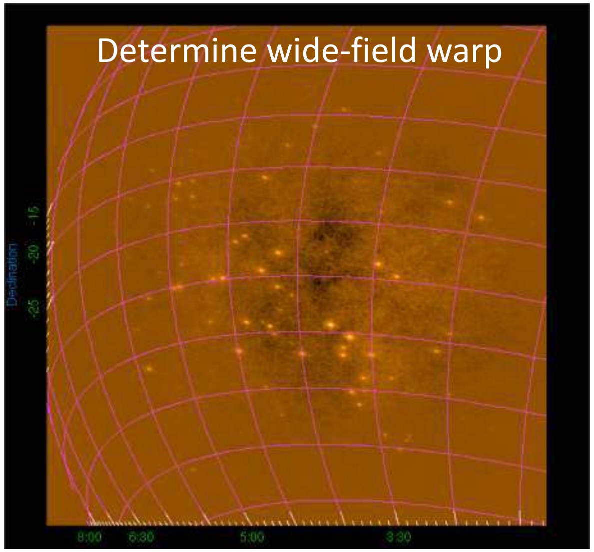
Constrained peeling → higher order models

SKA1-LOW Layout

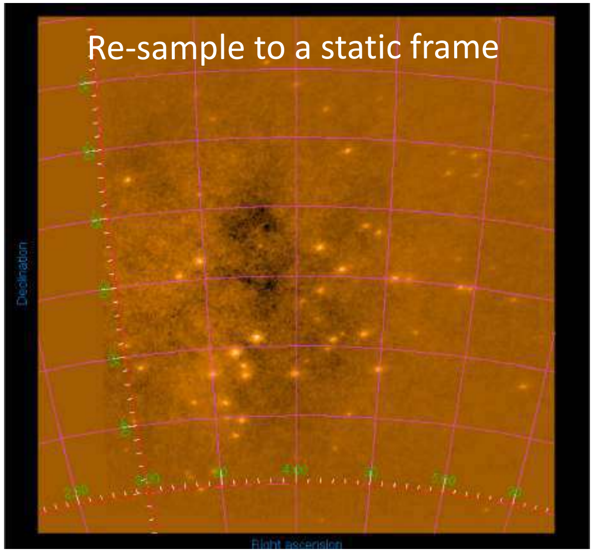


Warped Snapshots





Simulated data: field centre: -3.5 to +3.5 hrs



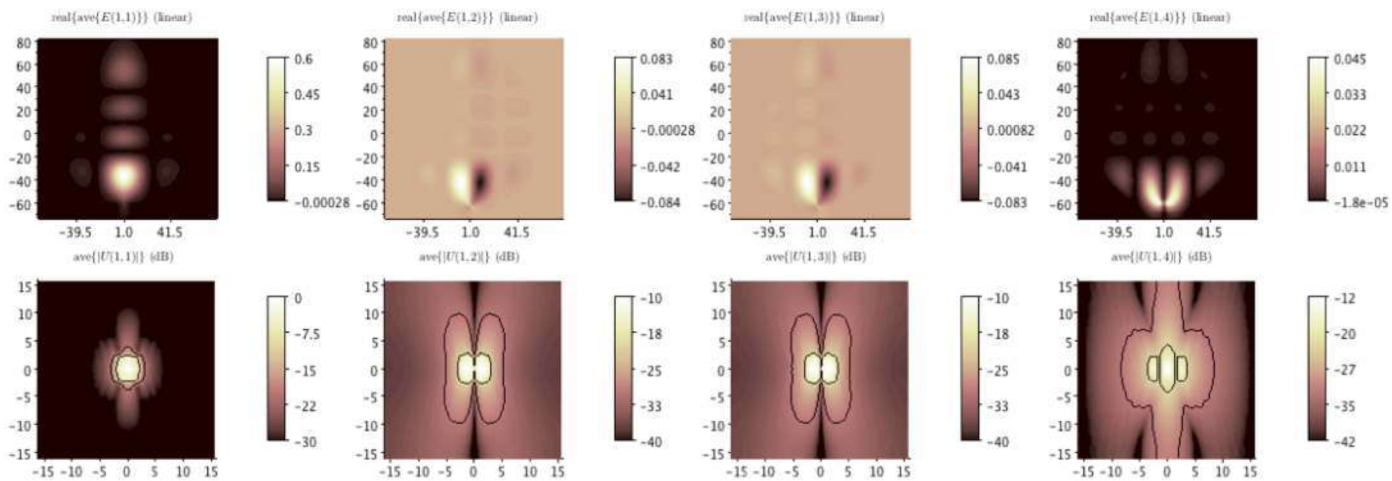
- Image warp is known and removed via image regridding
  - Potentially also the ionospheric perturbations.
- Time consuming to accurately calculate coordinates
  - Approximate: flat sky or interpolation
- Deep integrations occur in the image domain
  - With primary beam weighting, as in mosaicking

Variable Polarised Primary Beams

NN←xx NN←xy NN←yx NN←yy

All-sky polarised tile response

Fourier response



Variable Polarised Primary Beams

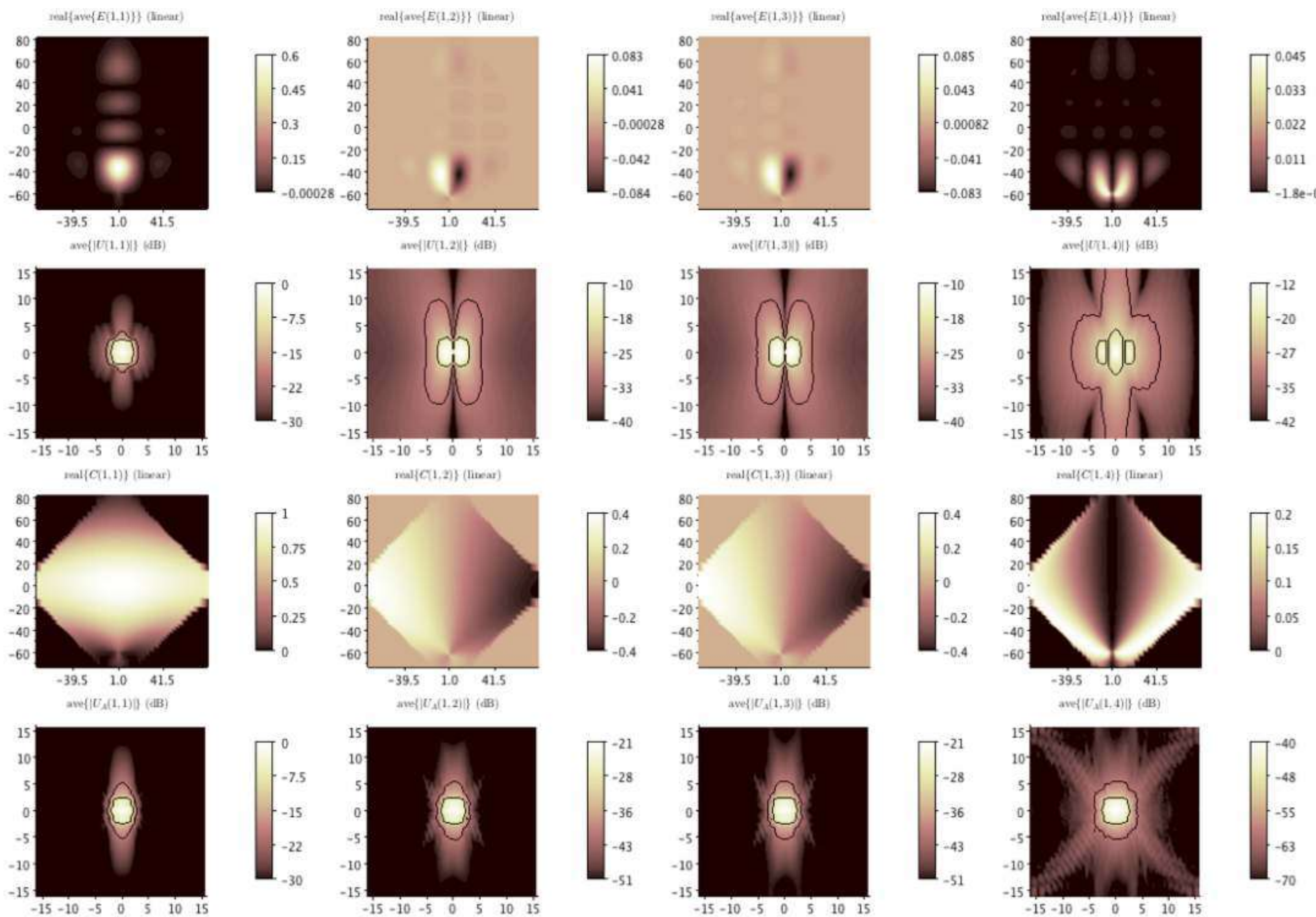
NN←xx | NN←xy | NN←yx | NN←yy

All-sky polarised tile response

Fourier response

Deal with curved sky in the image domain

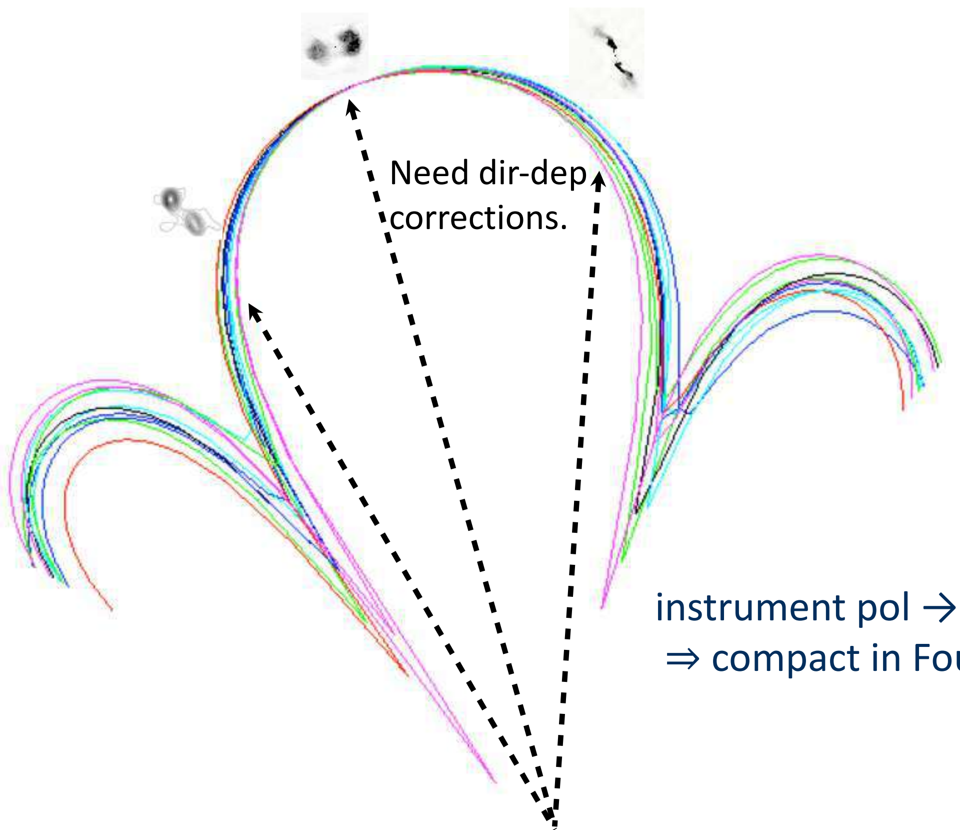
Remaining Instrument Fourier response



A projection

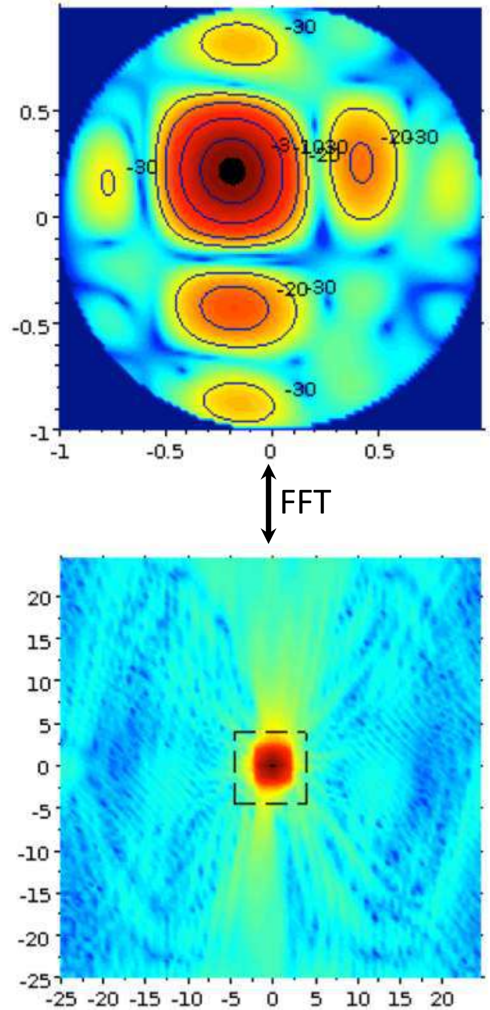
Apply direction-dependent corrections & weighting during gridding.

Not often used in practice, so still somewhat experimental.



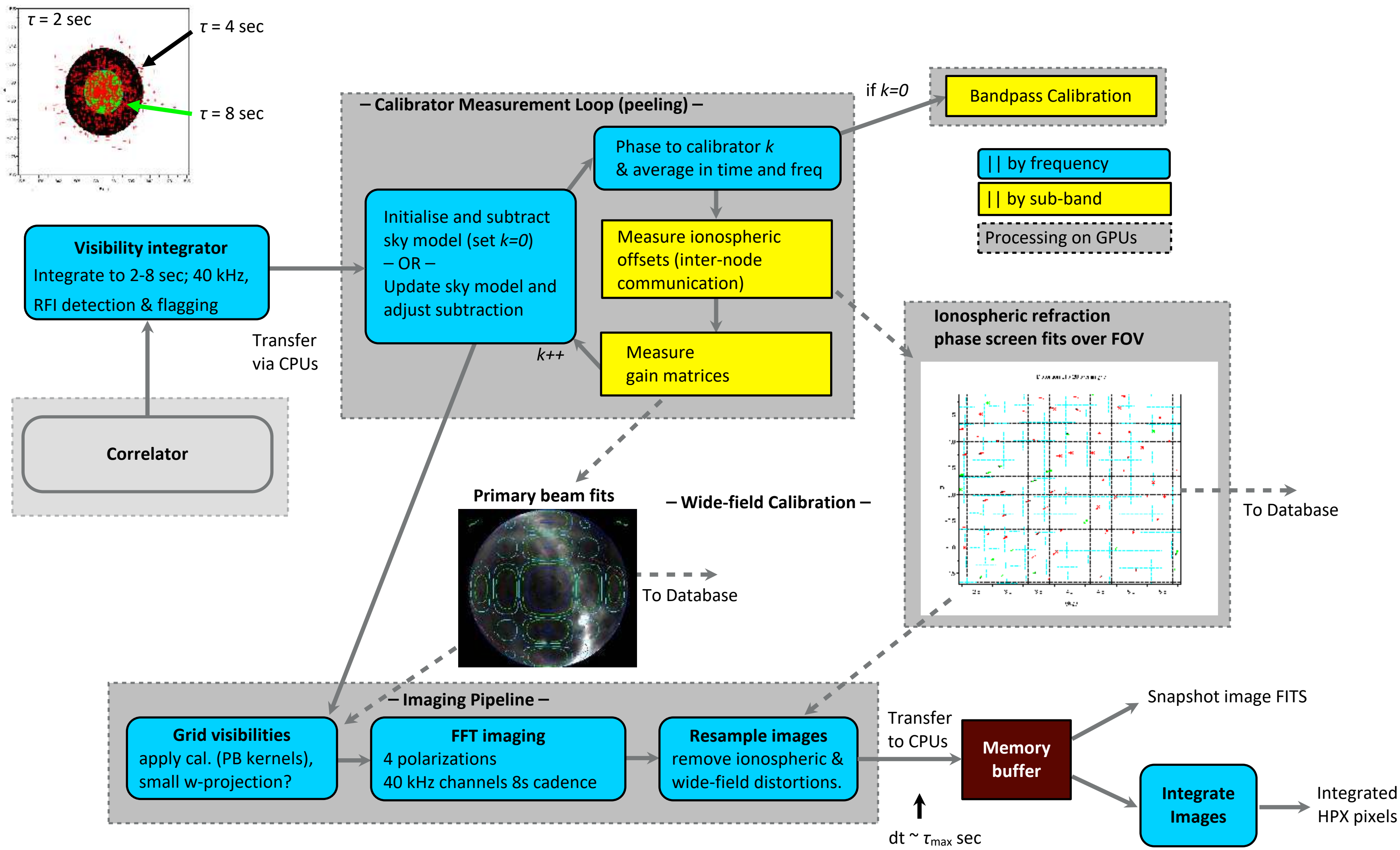
instrument pol → instrument frame
⇒ compact in Fourier space

Response of one pol to
unpolarised emission

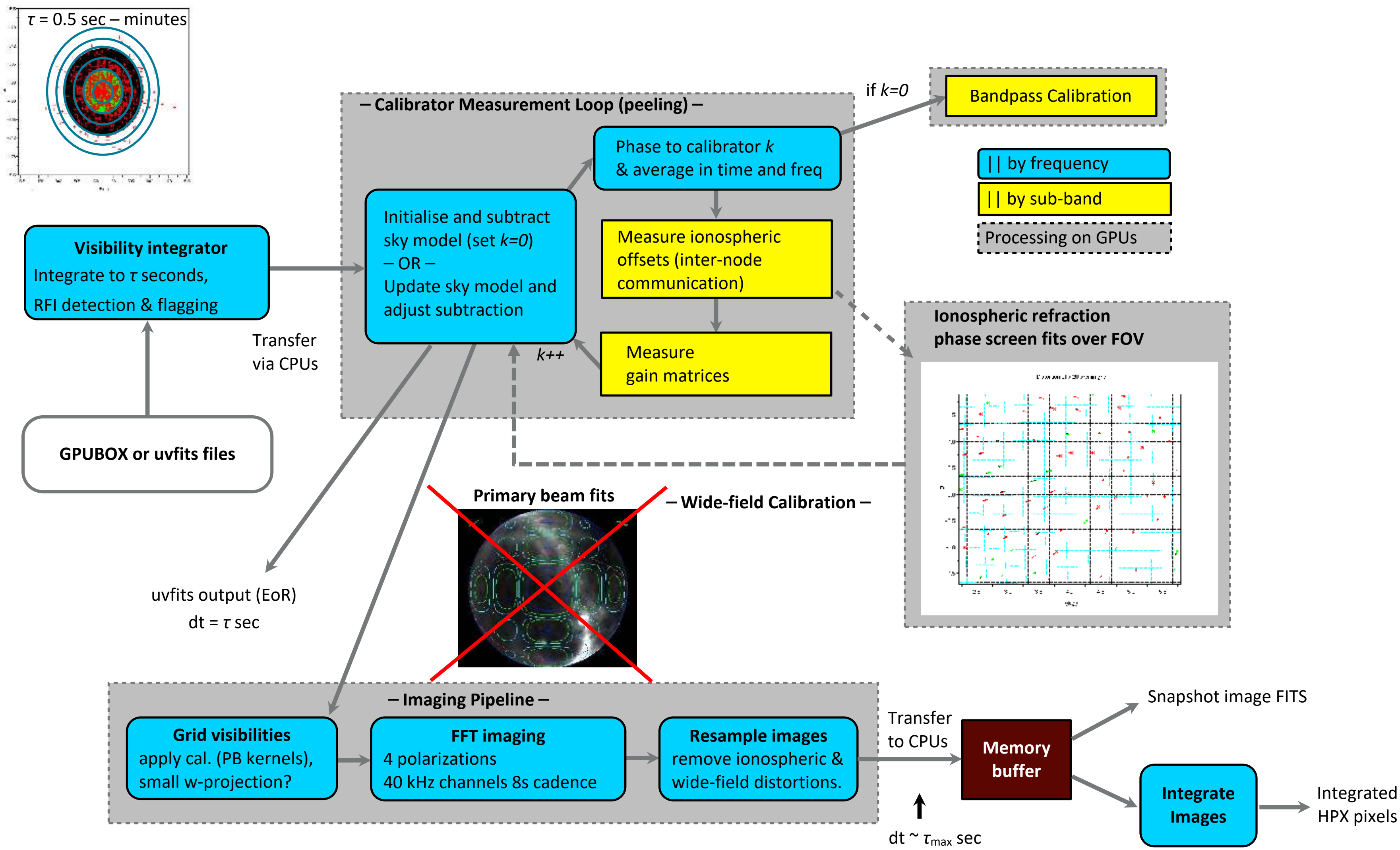


Data & Processing Flow

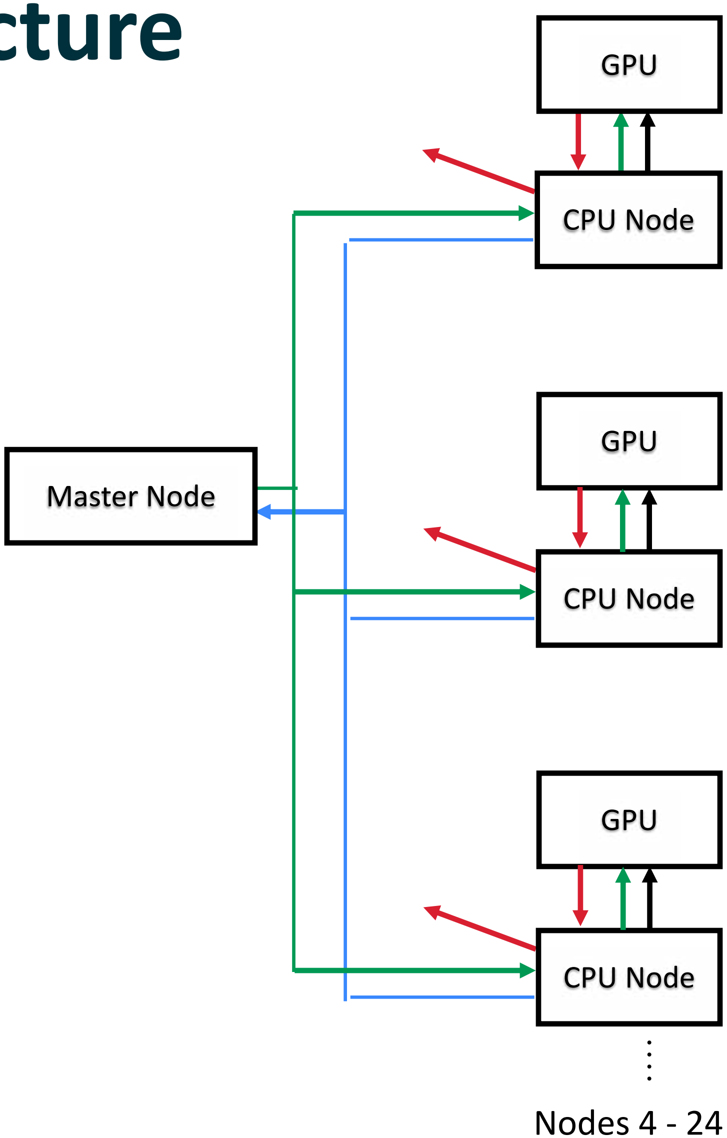
RTS Data Flow (original design)



RTS Data Flow (current design)



Heterogeneous Architecture



Galaxy@Pawsey 64 GPU nodes:

- Sandy Bridge EP CPUs
- K20X Kepler GPUs

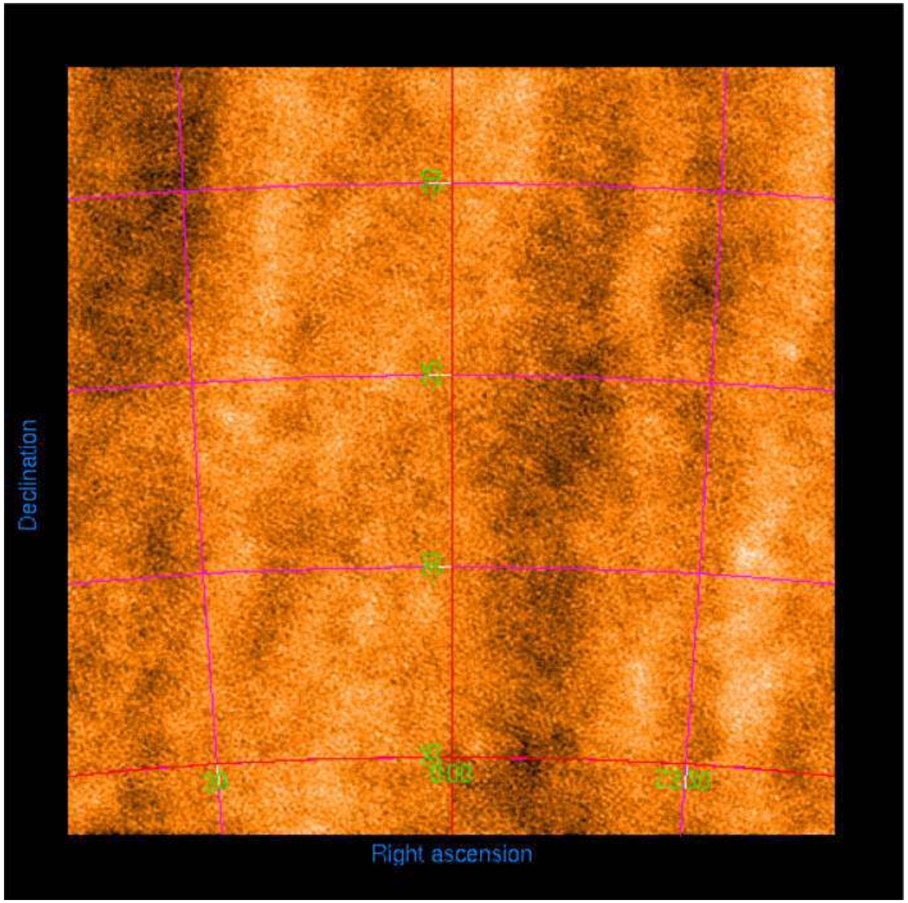
Typical Processing Steps

- Process each 2 – 5 minute obsid separately
- First run
  - Set the solution interval, τ_{max}, to be the full obsid (≈ 2 – 5 minutes).
  - Set the sky model to have a dominant calibrator that contains all components in the field of view.
  - Generate bandpass calibration solutions.
- Frequency / obsid consensus? (continuous, smooth solutions …)
- Second run
  - Set visibility integrator maximum to $\tau \approx 8$ seconds
  - Set sky model to have ≈ 1000 sources.
  - Subtract sky model with direction-dependent ionospheric calibration
  - Subtract a few sources with tile Jones matrix peeling if need be.
- Post-processing
  - Further image integration in time and/or frequency.
  - Conversion to Stokes images and FITS format.

Parameter Input Files

```
# for obsid 1061313984

ImportCotterBasename=../1061313984/1061313984
MetafitsFilename=../1061313984/1061313984

CorrDumpTime=0.5
CorrDumpsPerCadence=128
NumberOfIntegrationBins=8
NumberOfIterations=1

BaseFilename=../1061313984/*_gpubox

ObservationTimeBase=2456528.22648

ObservationFrequencyBase=167.035
ChannelBandwidth=0.04

calBaselineMin=20.0
calShortBaselineTaper=40.0
```

First Run



% rts_node_gpu config_file.in

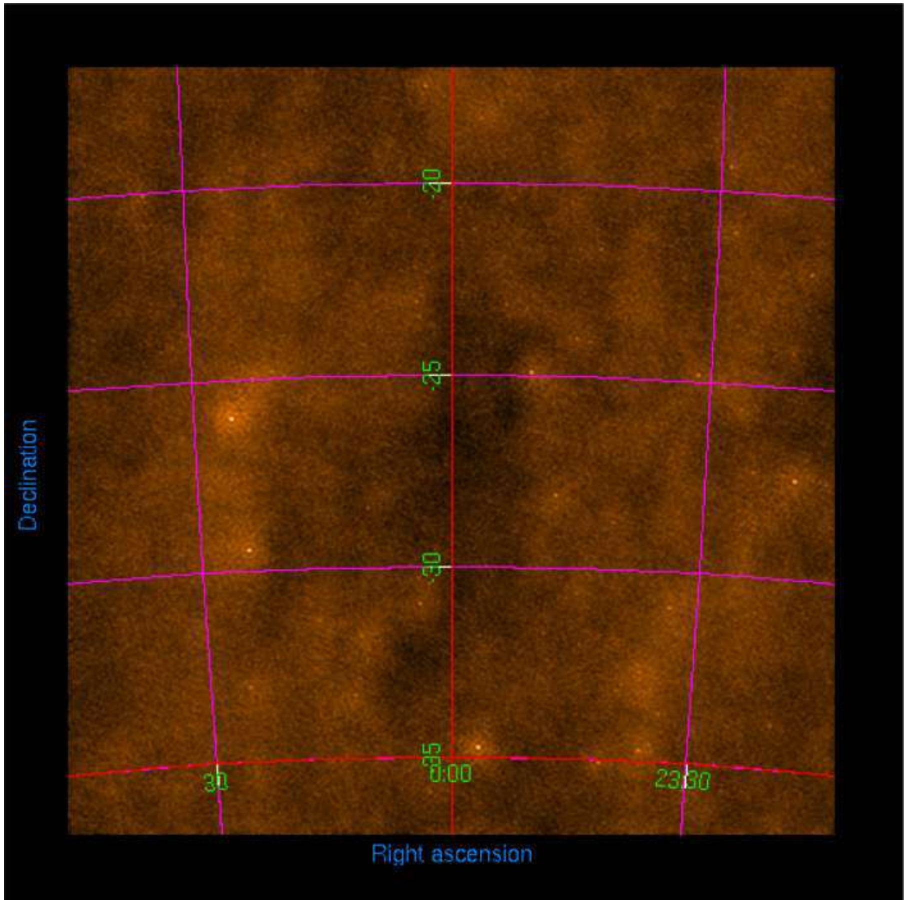
MakeImage=1
FieldOfViewDegrees=20
ImageOversampling=4

ObservationImageCentreRA=0.0
ObservationImageCentreDec=-27.0

FscrunchChan=32

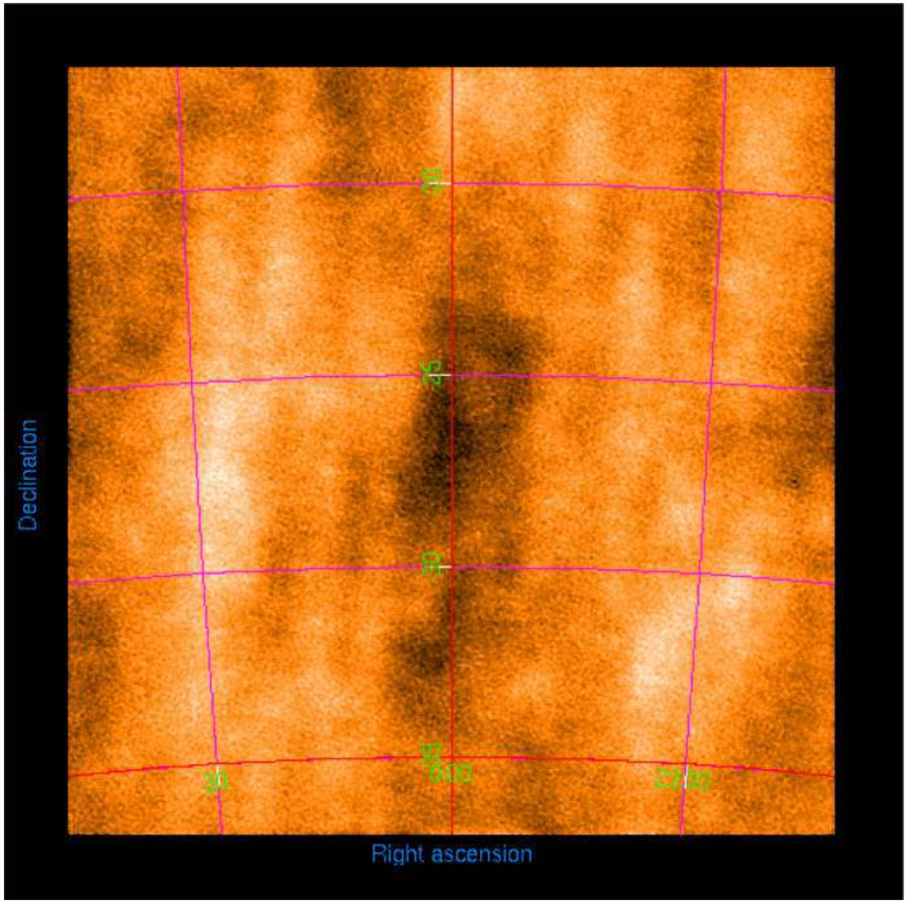
Run a single RTS worker node.
(process a single 1.28 MHz coarse channel)
Use “rts_gpu” for full MPI version.

First Run



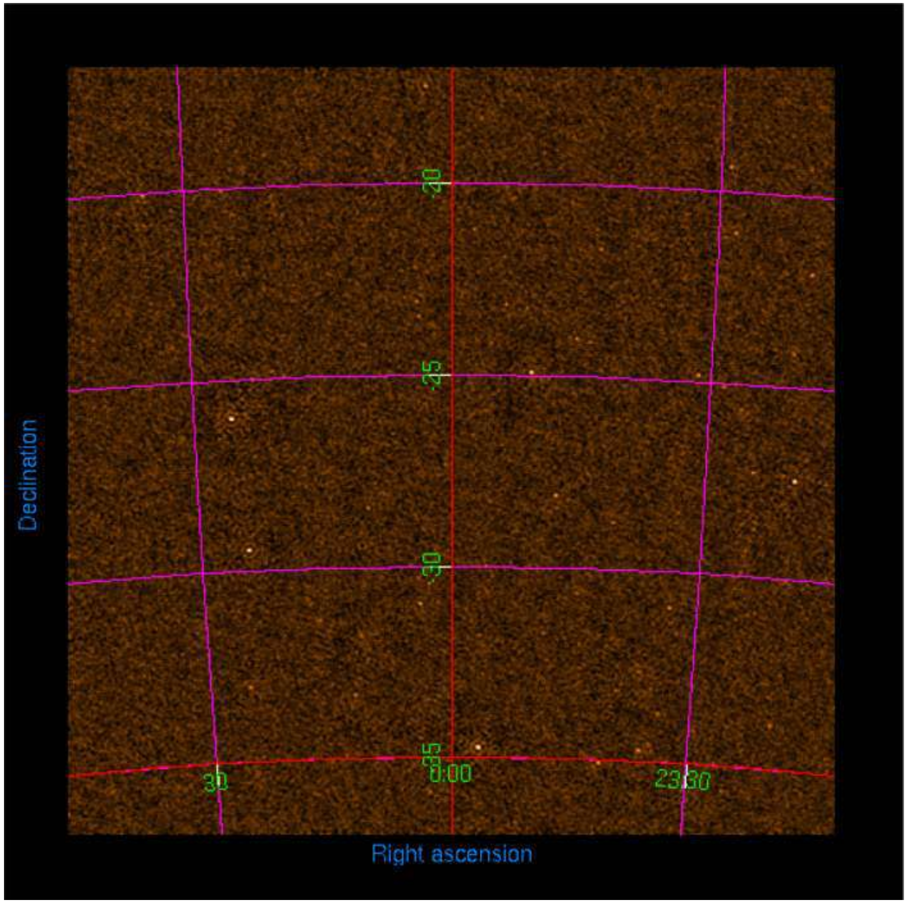

```
% python srclist_by_beam.py \
        -s srclist_puma-v2_complete.txt \
        -n 300 \
        -m ${obs}_metafits_ppds.fits

% rts_node_gpu config_file.in

        generateDIjones=1
        useStoredCalibrationFiles=0

        SourceCatalogueFile=patch300.txt

        NumberOfCalibrators=1
```

First Run



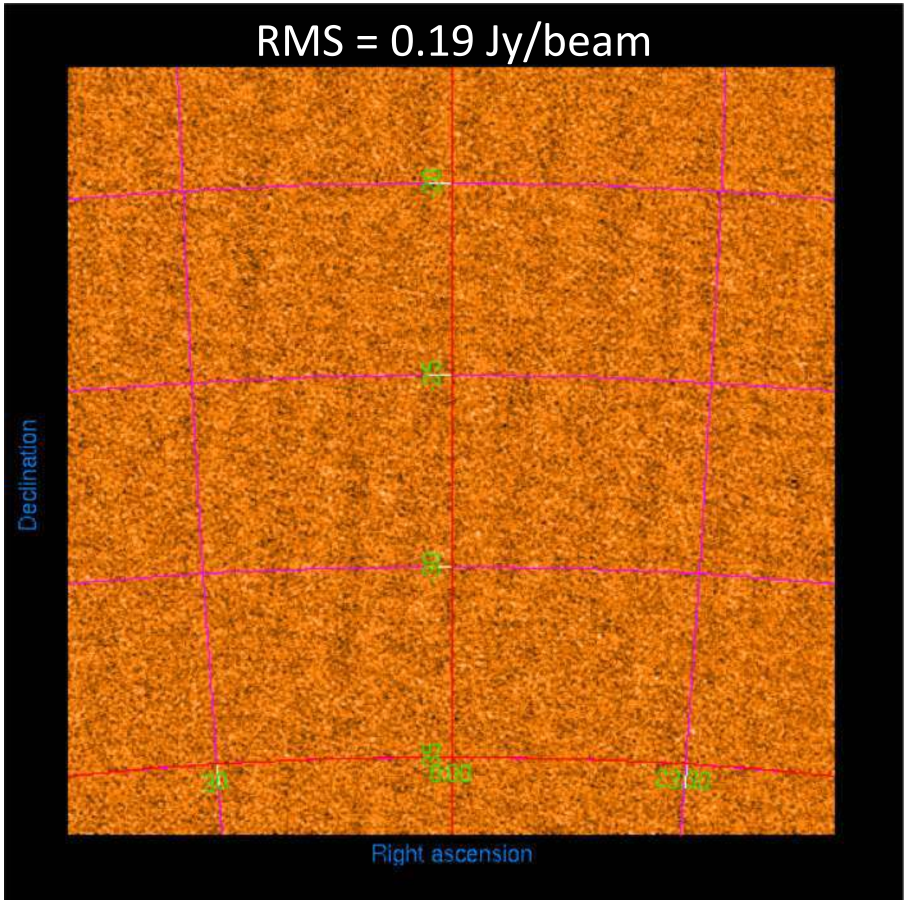

```
% python srclist_by_beam.py \
        -s srclist_puma-v2_complete.txt \
        -n 300 \
        -m ${obs}_metafits_ppds.fits

% rts_node_gpu config_file.in

        generateDIjones=1
        useStoredCalibrationFiles=0

        SourceCatalogueFile=patch300.txt

        NumberOfCalibrators=1
        NumberOfSourcesToPeel=1
```

First Run



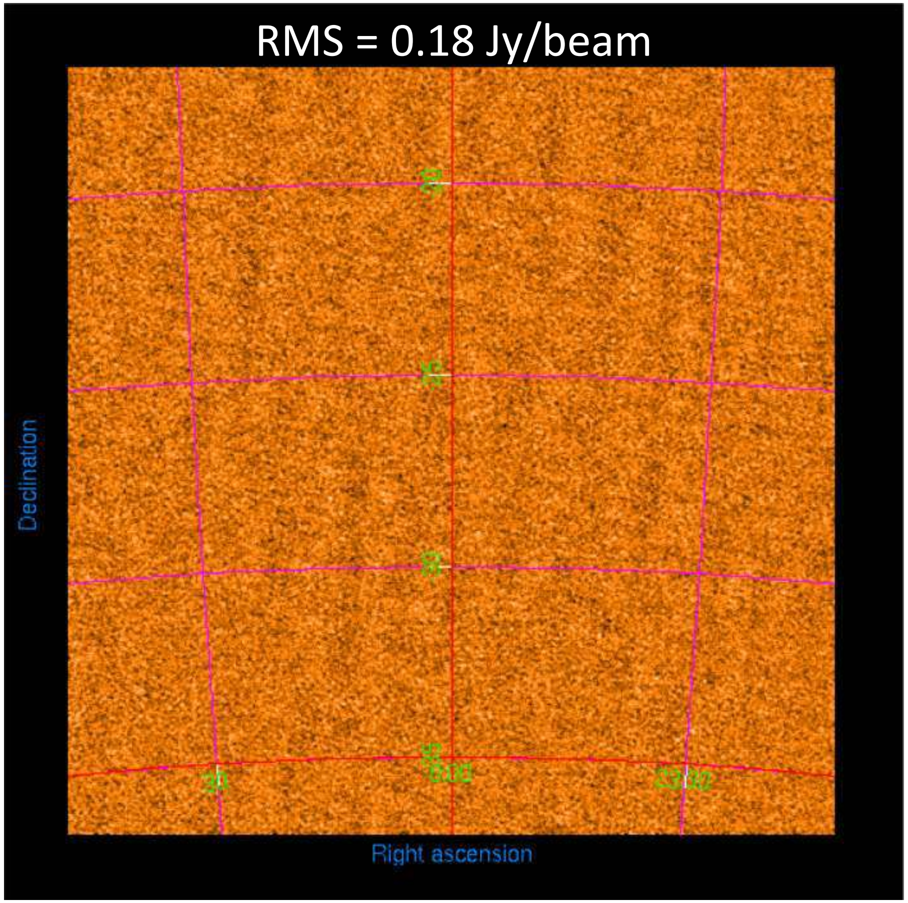

```
% python srclist_by_beam.py \
        -s srclist_puma-v2_complete.txt \
        -n 300 \
        -m ${obs}_metafits_ppds.fits

% rts_node_gpu config_file.in

        generateDIjones=1
        useStoredCalibrationFiles=0

        SourceCatalogueFile=patch300.txt

        NumberOfCalibrators=1
        NumberOfSourcesToPeel=0

        imgBaselineMin=20.0
        imgShortBaselineTaper=40.0
```

First Run




```
% python srclist_by_beam.py \
        -s srclist_puma-v2_complete.txt \
        -n 300 \
        -m ${obs}_metafits_ppds.fits

% rts_node_gpu config_file.in

        generateDIjones=1
        useStoredCalibrationFiles=0

        SourceCatalogueFile=patch300.txt

        NumberOfCalibrators=1
        NumberOfSourcesToPeel=1

        imgBaselineMin=20.0
        imgShortBaselineTaper=40.0
```

Second Run




```
% python srclist_by_beam.py \
        -s srclist_puma-v2_complete.txt \
        -o experimental -x \
        -n 300 \
        -m ${obs}_metafits_ppds.fits

% rts_node_gpu config_file.in

        generateDIjones=0
        useStoredCalibrationFiles=1

        SourceCatalogueFile=peel300.txt

        NumberOfSourcesToPrePeel=300
        NumberOfCalibrators=5
        NumberOfIonoCalibrators=300
        UpdateCalibratorAmplitudes=1
        NumberOfSourcesToPeel=300
```

Using the RTS

RTS Use

- EoR calibration and foreground (i.e. sky model) subtraction
- Calibration and imaging of Galactic polarisation
- Calibration and imaging for transient searches
- Calibration for pulsars beam-forming
- Calibration and imaging for other arrays (LEDA)

Running on Galaxy

slurm script:

```
#!/bin/bash -l
#SBATCH --job-name="RTS"
#SBATCH -o RTS-%A.out
#SBATCH --nodes=25
#SBATCH --ntasks-per-node=1
#SBATCH --time=00:20:00
#SBATCH --partition=gpuq
#SBATCH --account=mwaeor
#SBATCH --export=NONE
#SBATCH --mem=30000
#SBATCH --gres=gpu:1

module load rts

srun   --ntasks=25  --ntasks-per-node=1   --export=ALL   rts_gpu   rts_params.in
```

Parameter Input Files

```
# for obsid 1061313984

ImportCotterBasename=../1061313984/1061313984
MetafitsFilename=../1061313984/1061313984

CorrDumpTime=0.5
CorrDumpsPerCadence=128
NumberOfIntegrationBins=8
NumberOfIterations=1

BaseFilename=../1061313984/*_gpubox

ObservationTimeBase=2456528.22648

ObservationFrequencyBase=167.035
ChannelBandwidth=0.04

calBaselineMin=20.0
calShortBaselineTaper=40.0
```

Sky Catalogue Files

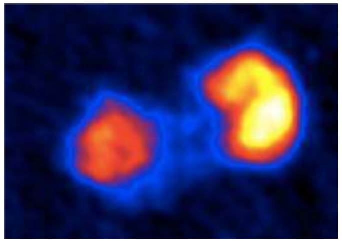
 # point source 1
SOURCE <name> ra_hrs dec_degs
FREQ freq_MHz I Q U V
FREQ freq_MHz I Q U V
...
ENDSOURCE

 # point source 2
SOURCE <name> ra_hrs dec_degs
FREQ freq_MHz I Q U V
...
ENDSOURCE
…

Sky Catalogue Files

 # Gaussian source
SOURCE <name> ra_hrs dec_degs
GAUSSIAN PA_degs major_arcmin minor_arcmin
FREQ freq_MHz I Q U V
ENDSOURCE

 # Shapelet source
SOURCE <name> ra_hrs dec_degs
FREQ freq_MHz I Q U V
SHAPELET PA_degs major_arcmin minor_arcmin
COEFF i1 j1 f1
COEFF i2 j2 f2
...
COEFF iN jN fN
ENDSOURCE



Fornax A
shapelet model

Sky Catalogue Files

 # Multi-component source
SOURCE <name> ra_hrs dec_degs
GAUSSIAN PA_degs major_arcmin minor_arcmin
FREQ freq_MHz I Q U V

COMPONENT ra_hrs dec_degs
FREQ freq_MHz I Q U V
ENDCOMPONENT

COMPONENT ra_hrs dec_degs
FREQ freq_MHz I Q U V
SHAPELET PA_degs major_arcmin minor_arcmin
COEFF i1 j1 f1
COEFF i2 j2 f2
...
COEFF iN jN fN
ENDCOMPONENT
ENDSOURCE

Flag Files

- Cotter flagging comes with the data.
- Extra flagging of tiles and/or frequency channels is available.
- Each MPI node will look for “flagged_tiles.txt” and “flagged_channels.txt” files and add extra flags.
- Very simple files:
  - single integer per line, representing to tile or channel to flag.
  - Integers start at zero and corresponds to input order
- Will be deprecate at some point, or advanced to contain metadata
  - But have been saying that for years, so mentioning here.

Summary

- RTS is very good at some things, and fast, but limited in scope

- To become a user, it is probably best to get in touch with me or one of the other groups using it:
  - EoR calibration and foreground (i.e. sky model) subtraction
  - Calibration and imaging of Galactic polarisation
  - Calibration and imaging for transient searches
  - Calibration for pulsars beam-forming
  - Calibration and imaging for other arrays (LEDA)