Modelling the SDP Imaging Pipeline

Anthony Griffin

Friday 13 February 2015

HIGH PERFORMANCE COMPUTING RESEARCH LAB
SCHOOL OF COMPUTER & MATHEMATCIAL SCIENCES

Disclaimers

- I’m not a radio astronomer. (I’m a signal processor)
- I’ve only been with AUT for a few weeks.
- I may say some wrong things.

Imaging Pipeline

- Major function of the Science Data Processor (SDP)

- Takes the output of the CSP (Central Signal Processor)
  - Visibilities (uv-plane)
  - (Measurements in the Fourier domain)
- Processes the Visibilities
- Produces an image of a region of the sky

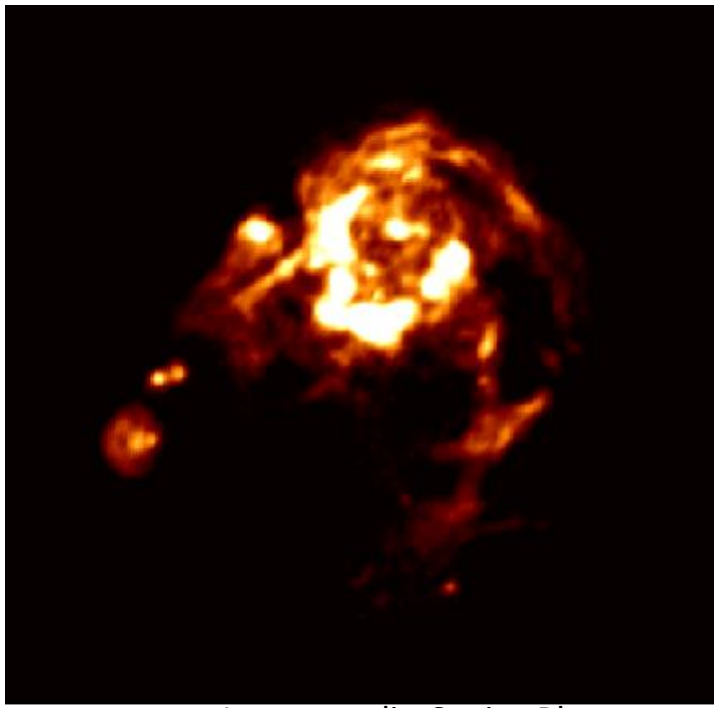
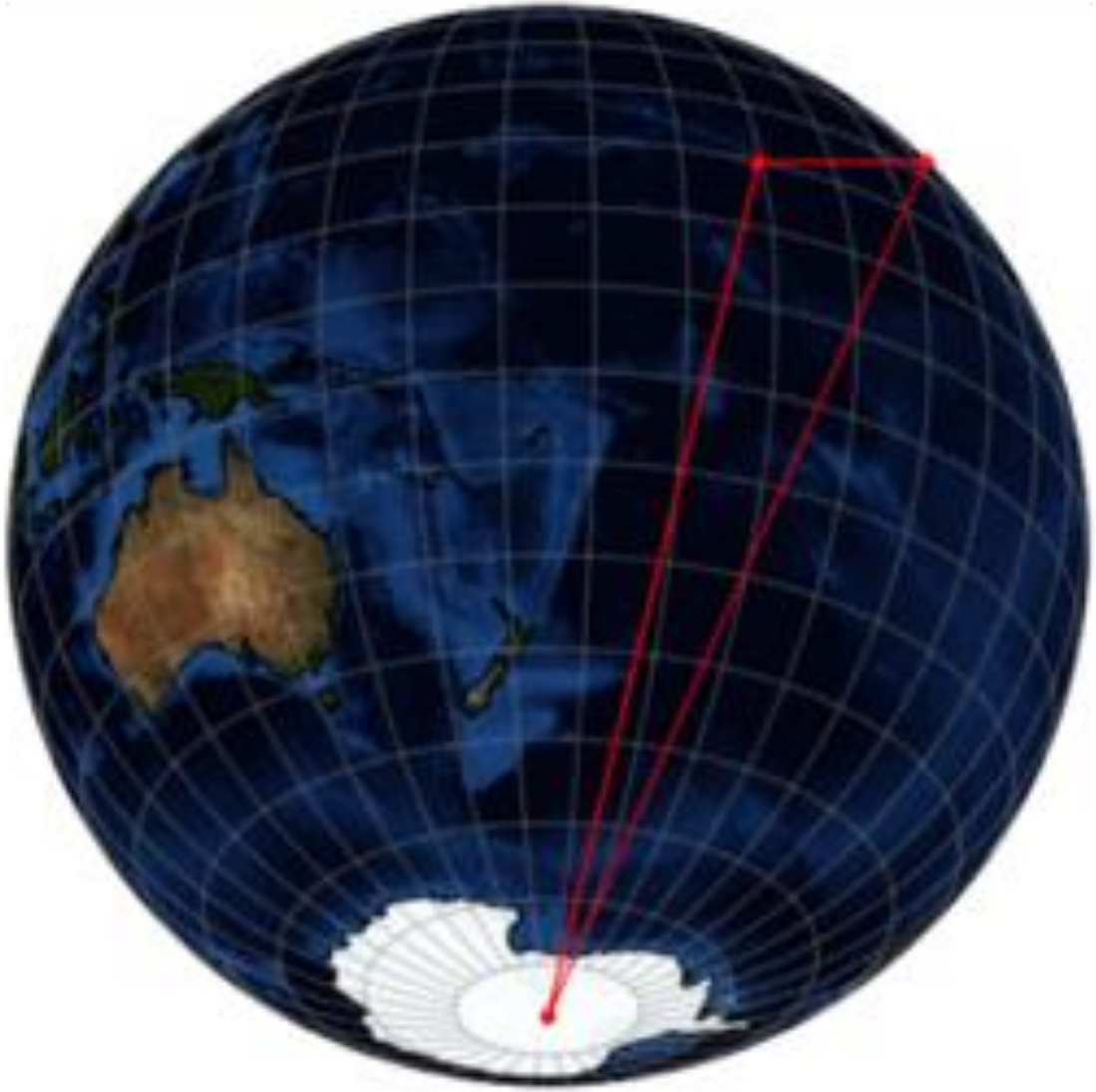


Image credit: Sanjay Bhatnagar

Imaging Pipeline

- Broken up into three main blocks:
  - Gridding
  - IFFT
  - Deconvolution

Visibilities



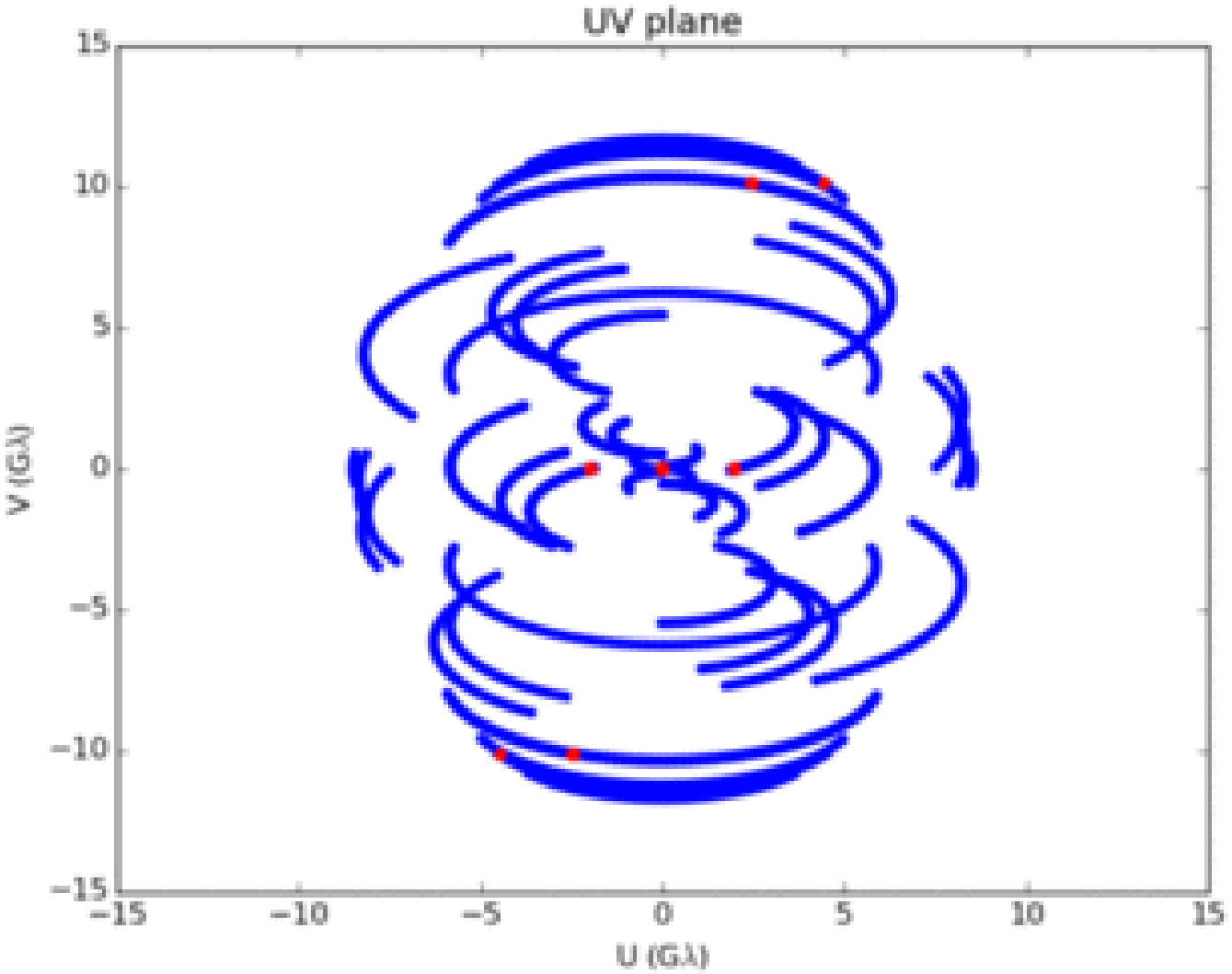


Image credits: Laura Vertatschitsch

Visibilities

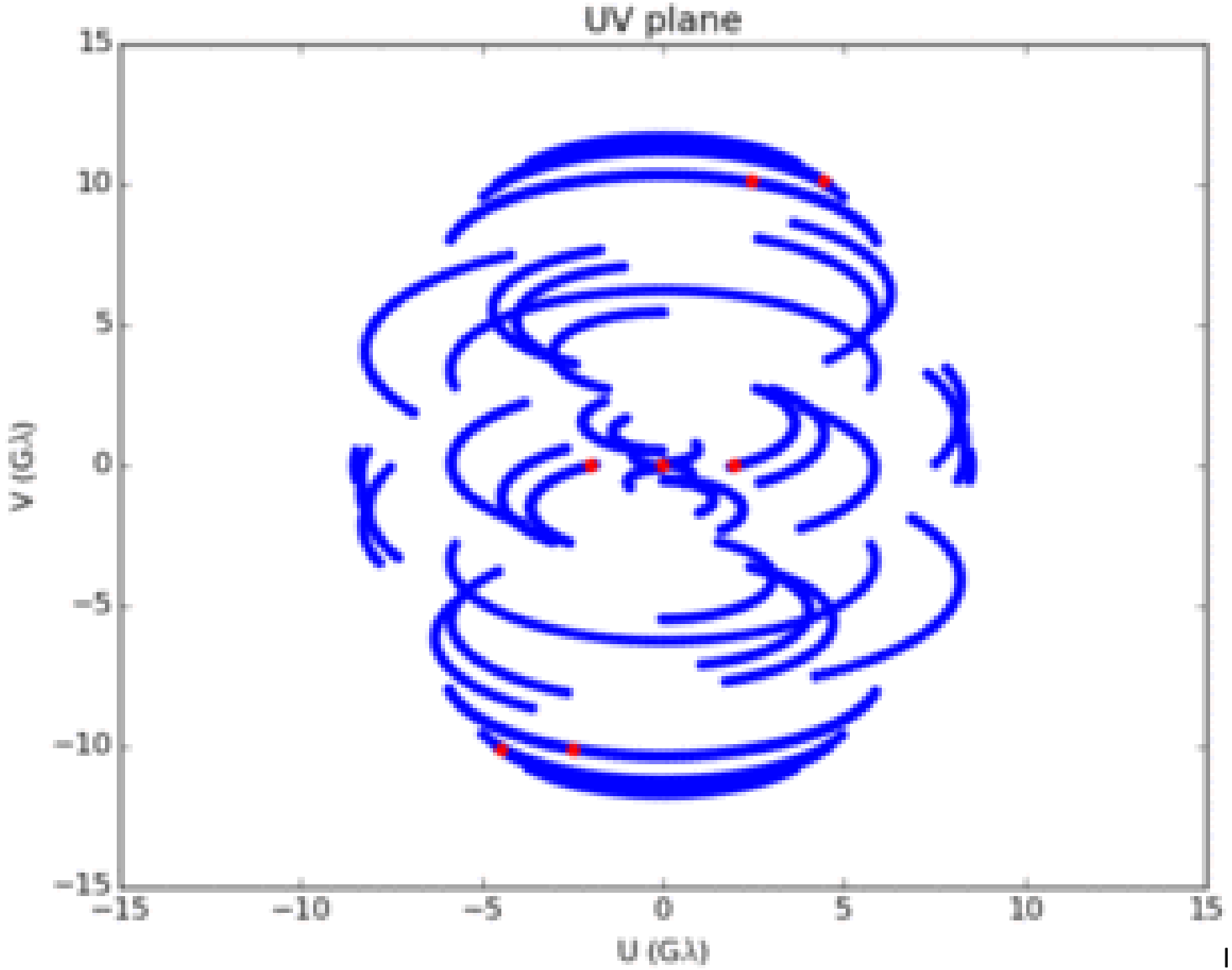


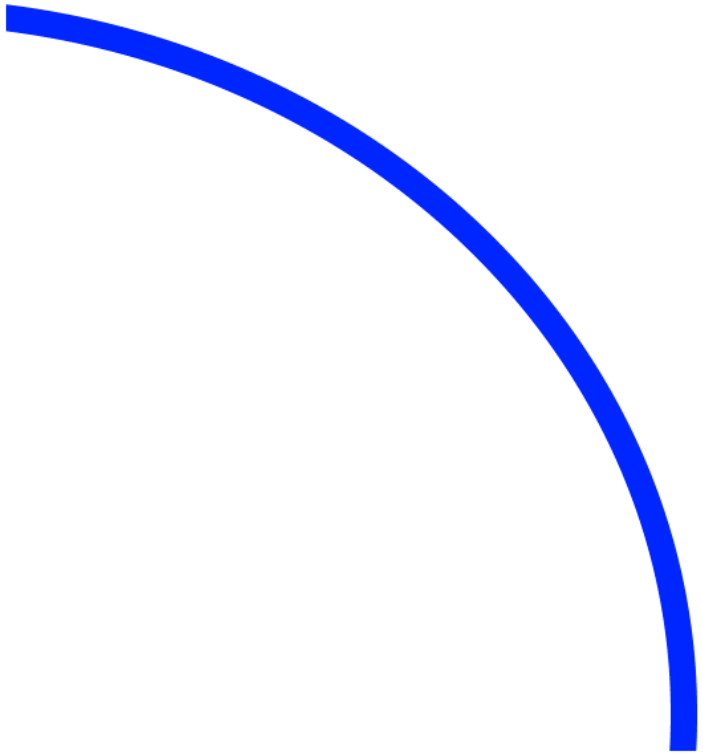
Image credits: Laura Vertatschitsch

Gridding

- Visibilities in uv-plane

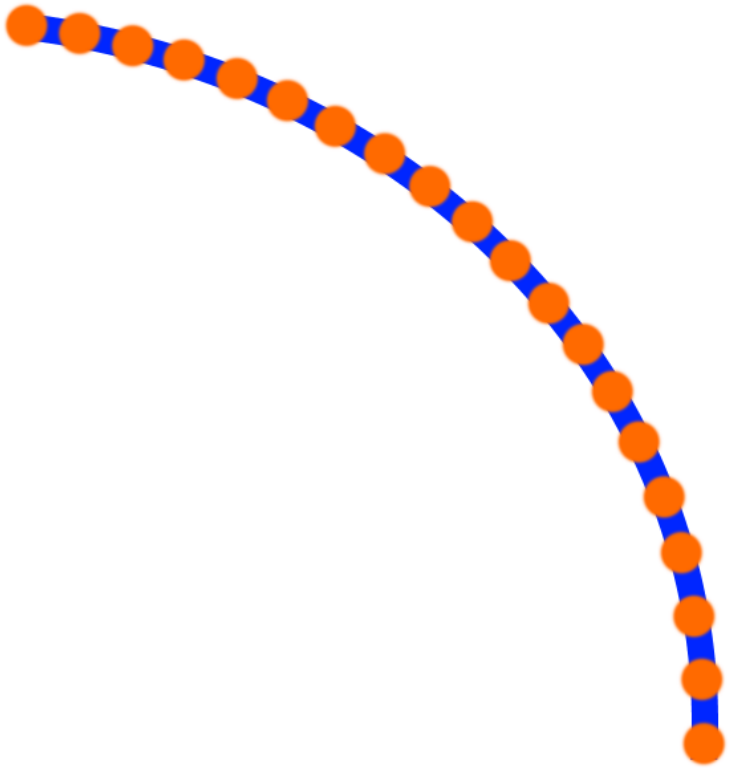
Gridding

- Visibilities in uv-plane



Gridding

- Visibilities in uv-plane
- Sampled in “polar” coordinates

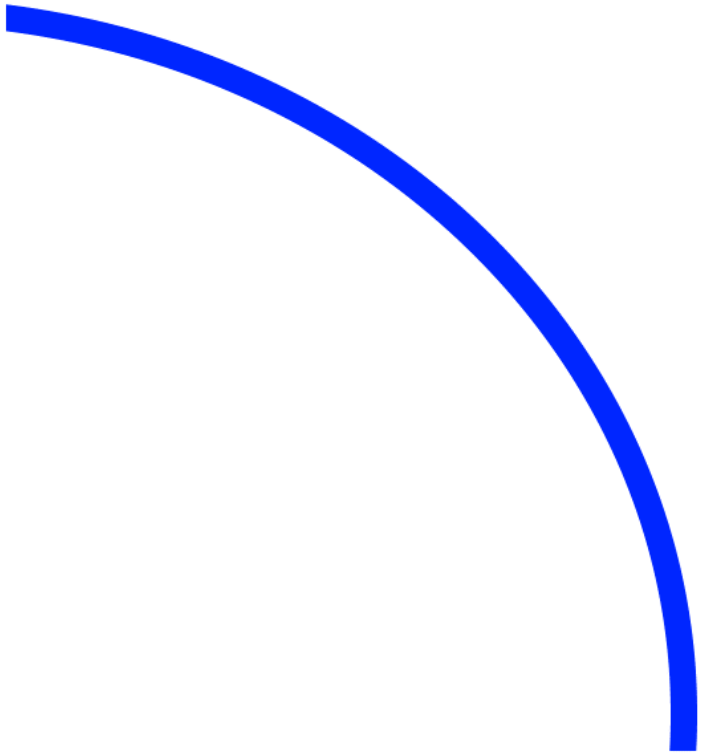


Gridding

- Visibilities in uv-plane
- Sampled in “polar” coordinates

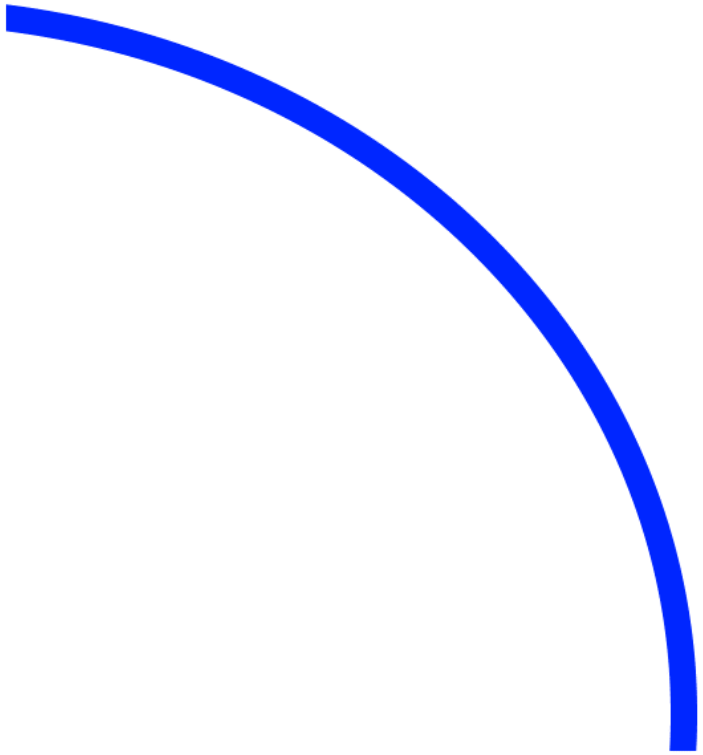
Gridding

- Visibilities in uv-plane
- Sampled in “polar” coordinates



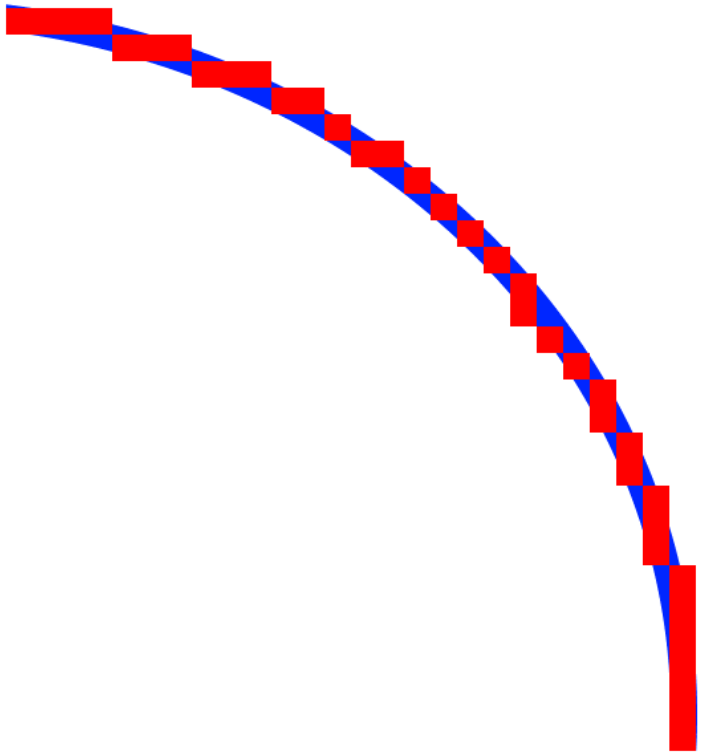
Gridding

- Visibilities in uv-plane
- Sampled in “polar” coordinates
- Need them in Cartesian coordinates for IFFT



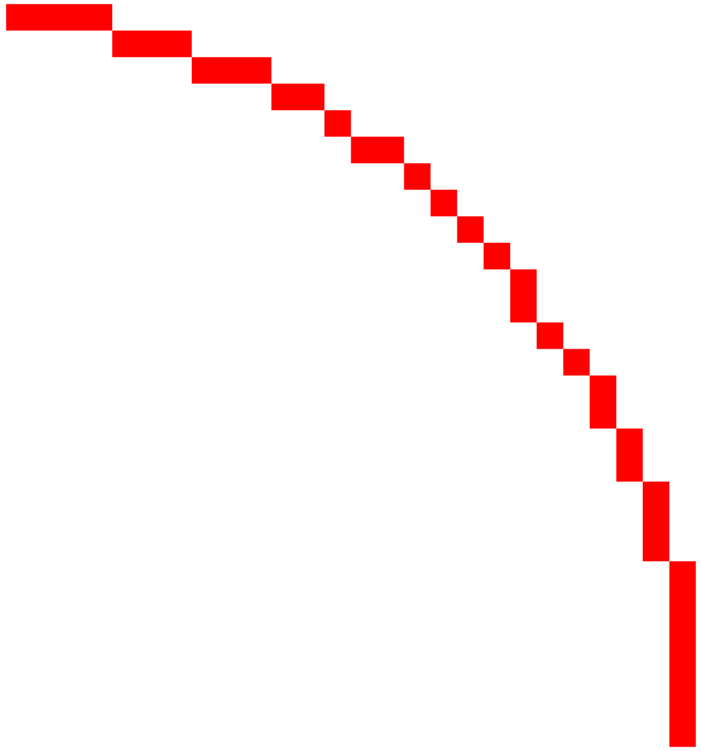
Gridding

- Visibilities in uv-plane
- Sampled in “polar” coordinates
- Need them in Cartesian coordinates for IFFT
- Nearest neighbour



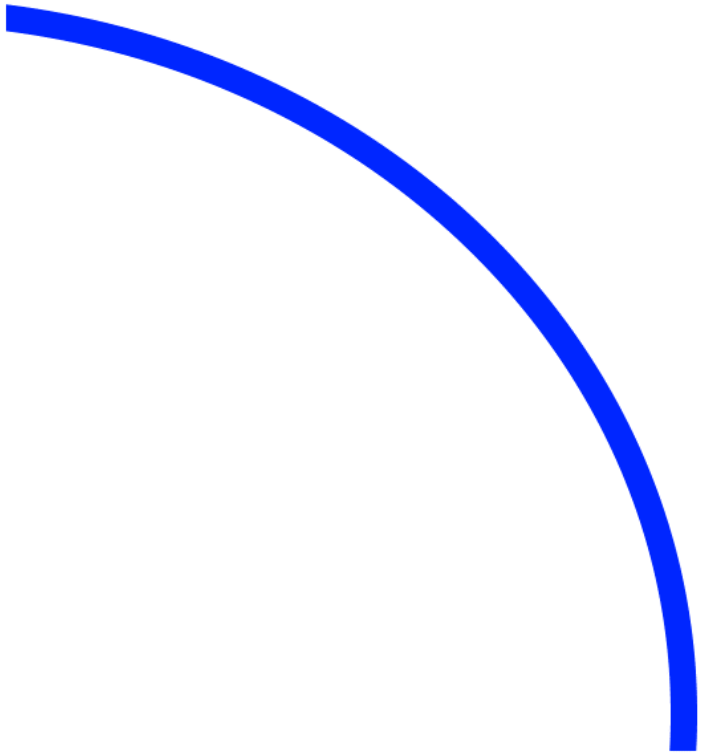
Gridding

- Visibilities in uv-plane
- Sampled in “polar” coordinates
- Need them in Cartesian coordinates for IFFT
- Nearest neighbour



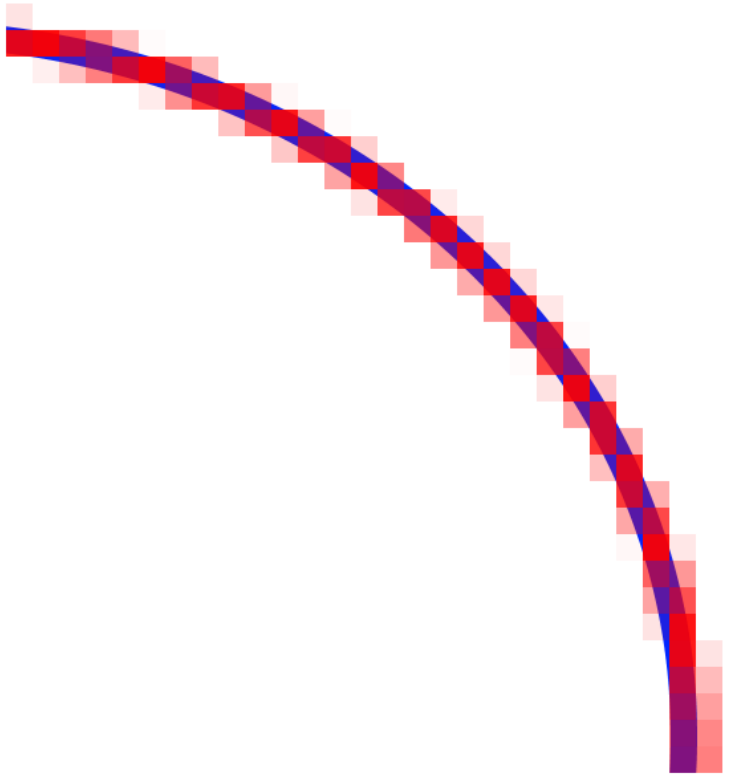
Gridding

- Visibilities in uv-plane
- Sampled in “polar” coordinates
- Need them in Cartesian coordinates for IFFT
- Nearest neighbour



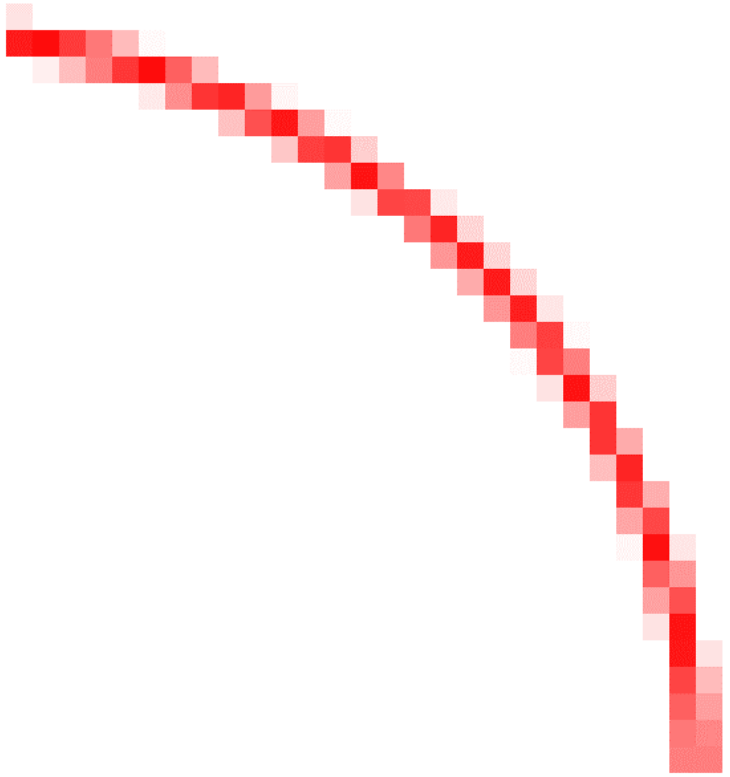
Gridding

- Visibilities in uv-plane
- Sampled in “polar” coordinates
- Need them in Cartesian coordinates for IFFT
- Nearest neighbour
- Convolution with a kernel (“anti-aliasing”)



Gridding

- Visibilities in uv-plane
- Sampled in “polar” coordinates
- Need them in Cartesian coordinates for IFFT
- Nearest neighbour
- Convolution with a kernel (“anti-aliasing”)



Gridding

- Visibilities in uv-plane
- Sampled in “polar” coordinates
- Need them in Cartesian coordinates for IFFT
- Nearest neighbour
- Convolution with a kernel (“anti-aliasing”)
- Many visibilities, 40% of SDP processing

What's actually happening

True Visibilities

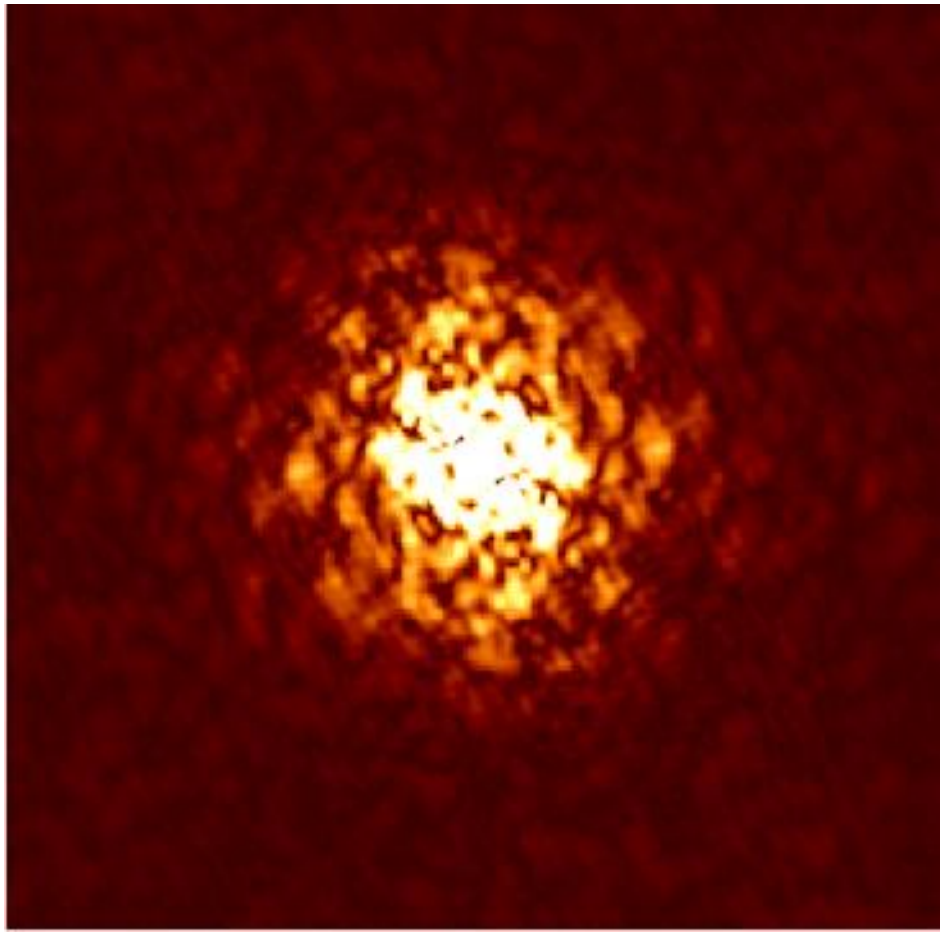
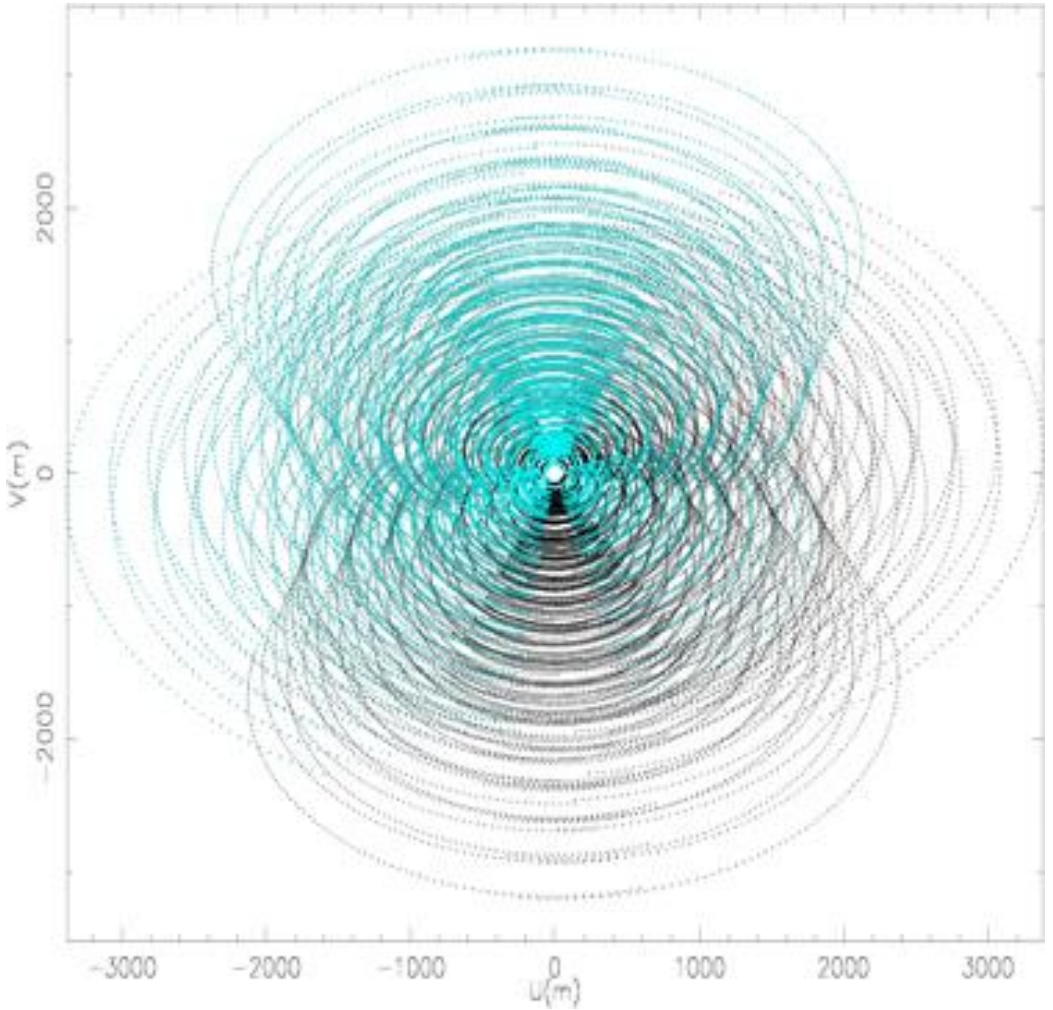


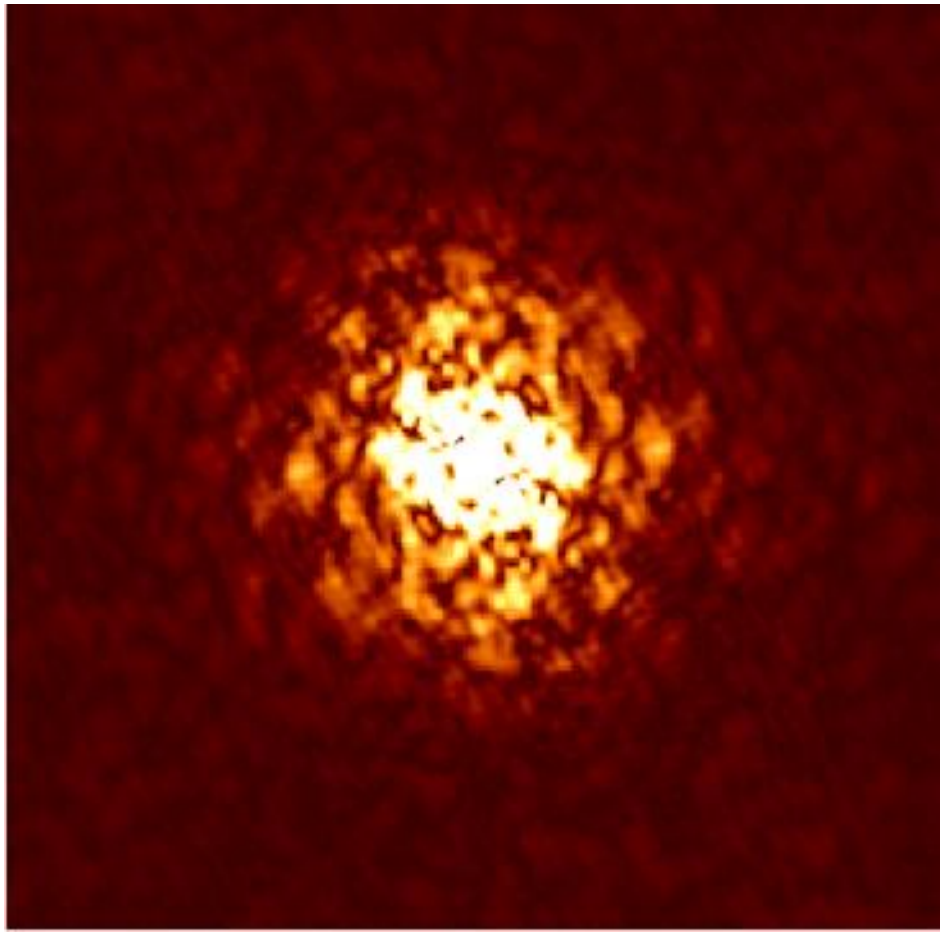
Image credits: Sanjay Bhatnagar

Sampling Function

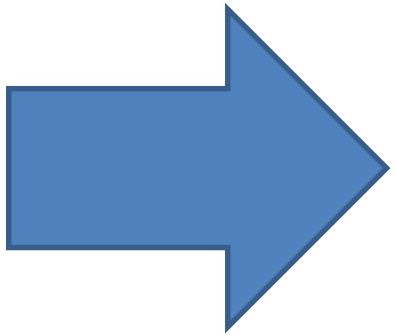


What's actually happening

True Visibilities



Sampling



Sampled Visibilities

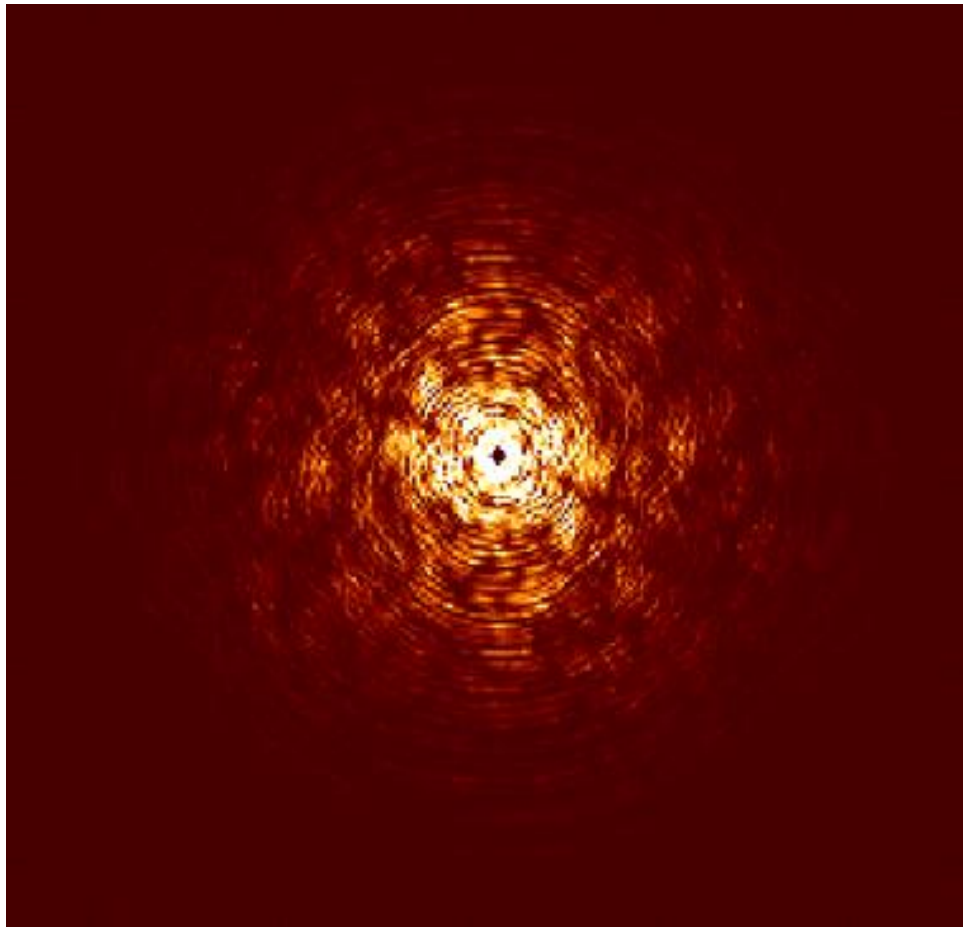
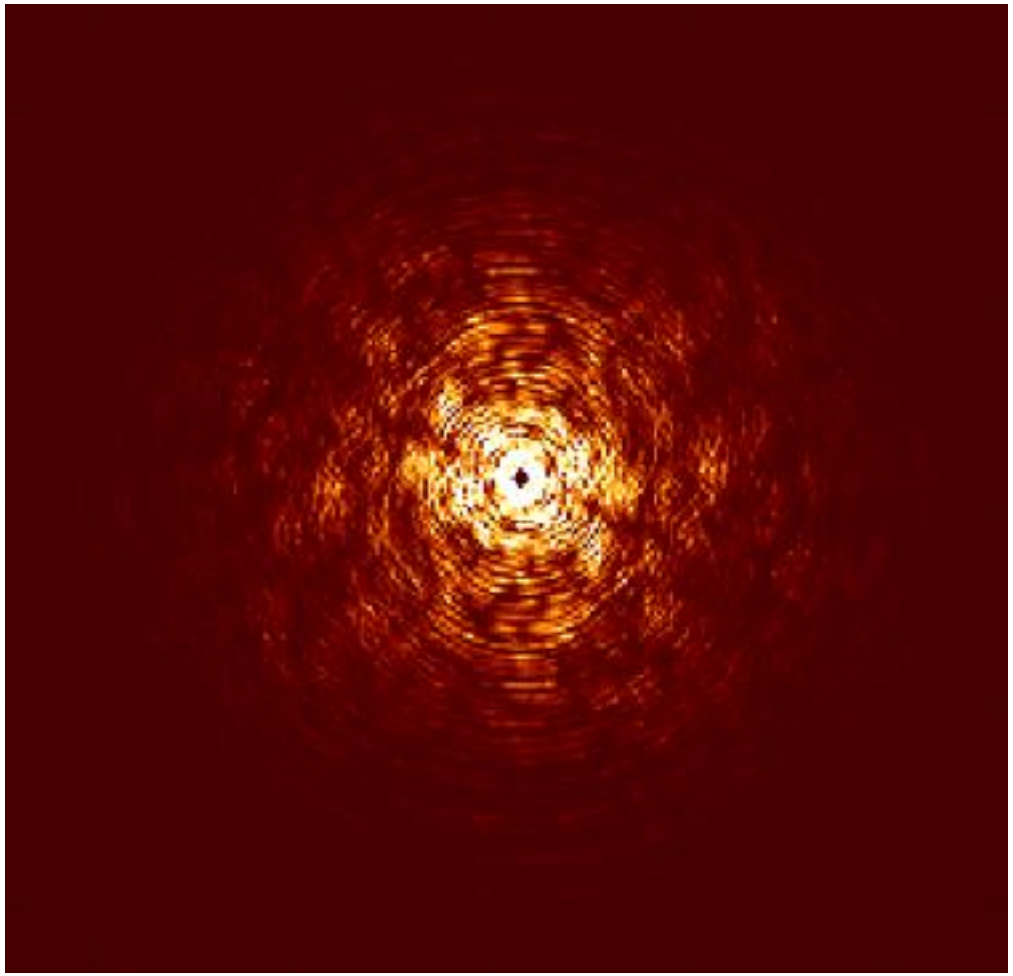


Image credits: Sanjay Bhatnagar

Backprojection

Sampled Visibilities



IFFT

Dirty Image

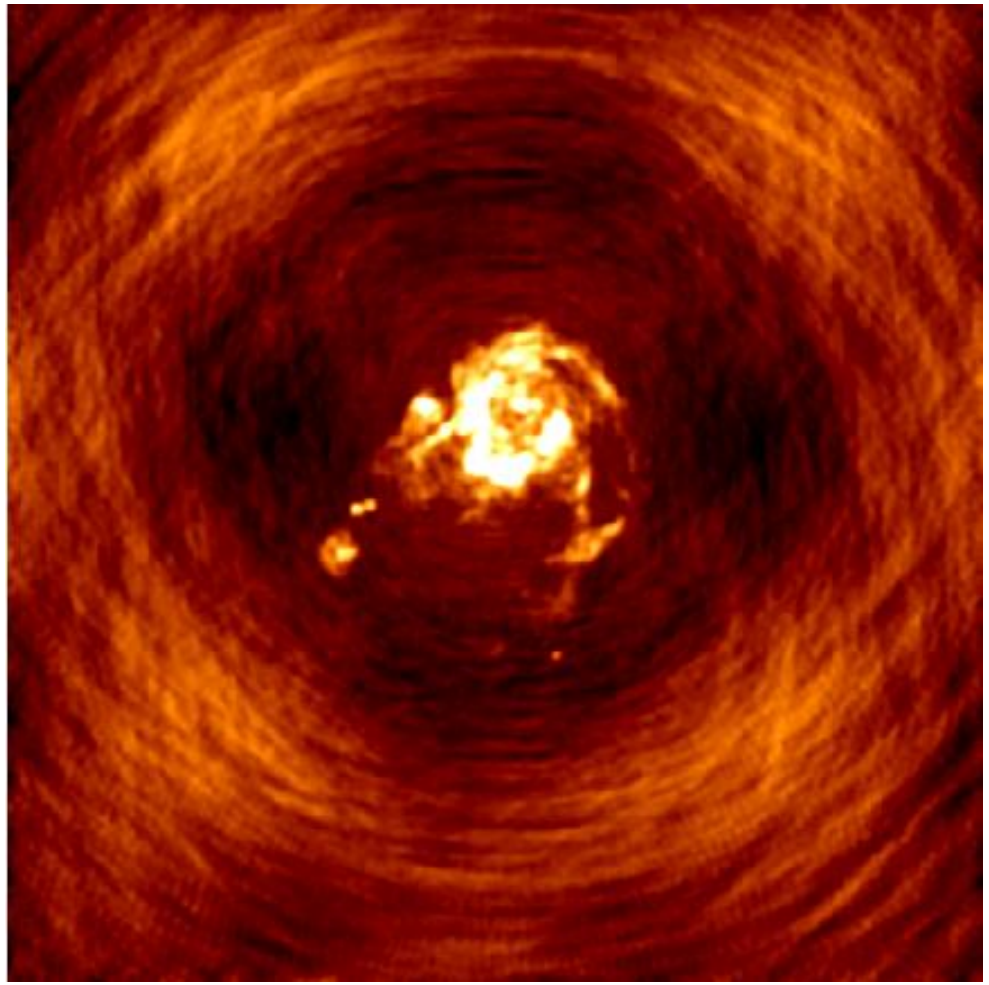


Image credits: Sanjay Bhatnagar

IFFT

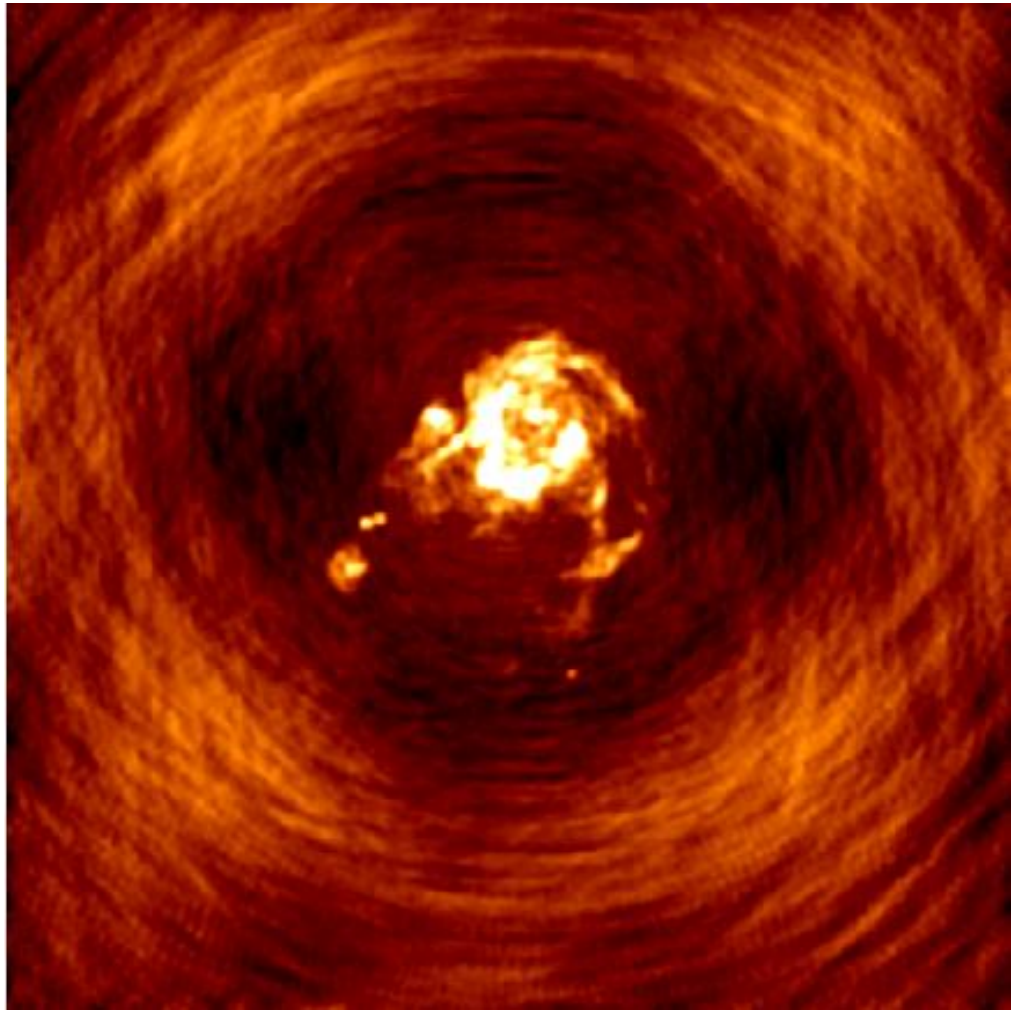
- Classic, well understood algorithm
- At least 16K by 16K (or even 60K by 60K)
  - very big
- 40% of SDP processing

Deconvolution

- This removes the effect of the under-sampling in the uv-plane
- Uncovers the “true” sky map
- Many different deconvolution algorithms
- CLEAN is the classic algorithm
  - Högbom, Clark, Cotton-Schwab, other variants
- Sparse Reconstruction and Compressed Sensing methods are the most recent

Deconvolution

Dirty Image



CLEANing

Clean Image

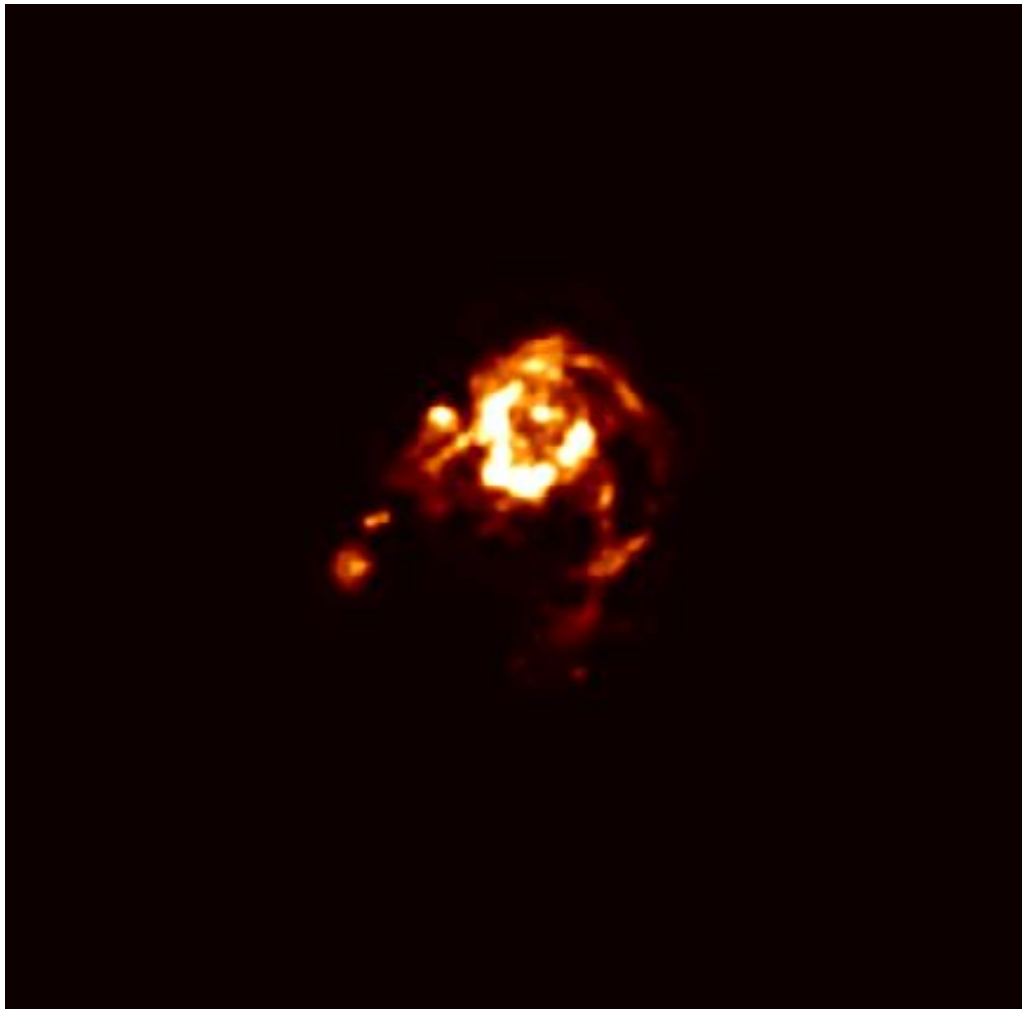


Image credits: Sanjay Bhatnagar

Deconvolution (cont.)

- This deconvolution is more an “art” than a science
  - When do we stop CLEANing?

- It is normally performed by a skilled operator
- In the SKA this will have to be done in an unsupervised manner
- Basic CLEAN takes 20% of the SDP processing time

Why model?

- To give us a reference model for implementation variations
- To allow us to try out various imaging algorithms
- To determine the required mathematical precision
  - Double- or single-precision floating point
  - (64- or 32-bit)

Double-Precision Floating Point

- Gives the best algorithmic performance
- This is how all deconvolution imaging has been done to date
  - AFAIK!
- However, a lot of processors don’t support double precision

Single-Precision Floating Point

- Many benefits:
  - lower power (less than half)
  - lower cost (less than half)
  - reduced storage
  - reduced communication
  - higher throughput, for example, NVIDIA Tesla K40 GPU
    - DP 1.43 TFLOPS vs SP 4.39 TFLOPS (3x increase)
- But possibly with reduced algorithmic performance??

Modelling

- MATLAB
  - Fixed-point ??
- Simulink

What do we need?

- Input Data
  - Visibilities (OK)
- Output Data
  - Images (cleaned references corresponding to the data)
- Advice
  - about the deconvolution algorithms to consider
  - about the kind of data to consider (noise level)