Imaging and Deconvolution

When in doubt, try a Fourier Transform

Rhodes University RATT Group Introduction Lectures 2014

We usually build/use interferometric arrays as a round-about way of making an image of the sky. Given the visibilities, which sample the spatial frequency domain, we want to produce an model, in the spatial domain, that is as close to the true sky as possible. To do this, generally we go through some process akin to:

- gridding (skipping this, long path for small return)
- signal conditioning: weights and tapers
- deconvolution cycle:
  - Fourier transform
  - Partial sky model extraction
  - Inverse-Fourier transform
  - Partial sky model removal
- output: final sky model and residuals

The Fourier Transform is the most important transform/equation/bit of mathematics of the modern day.

2D Fourier Transform

$$I(l,m) \rightleftharpoons V(u,v)$$

$$I(l,m) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} V(v,u)e^{-2\pi i(ul+vm)}\ du\ dv$$

By Euler's formula:

$$I(l,m) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} V(v,u)(\cos(2\pi(ul+vm) - i\sin(2\pi(ul+vm)))\ du\ dv$$

The Fourier Transform decomposes an image in different sine and cosine spatial scales.

The inverse Fourier Transform is the same as the Fourier Transform (with a phase flip). This is one of the many powers of the Fourier Transform

$$V(u,v) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} I(l,m)e^{2\pi i(ul+vm)}\ dl\ dm$$

The Fast Fourier Transform (FFT)

2D Discrete Fourier Transform (DFT)

in radio astronomy there is a closely related equation termed the 'Direct' Fourier Transform

$$I(l,m) = \frac{1}{N_u N_v} \sum_{u=0}^{N_u-1} \sum_{v=0}^{N_v-1} V(u,v) e^{-2\pi i(\frac{ul}{N_u}+\frac{vm}{N_v})}\ du\ dv$$

Makes the samples discrete, but the function becomes periodic at the spatial Nyquist frequency

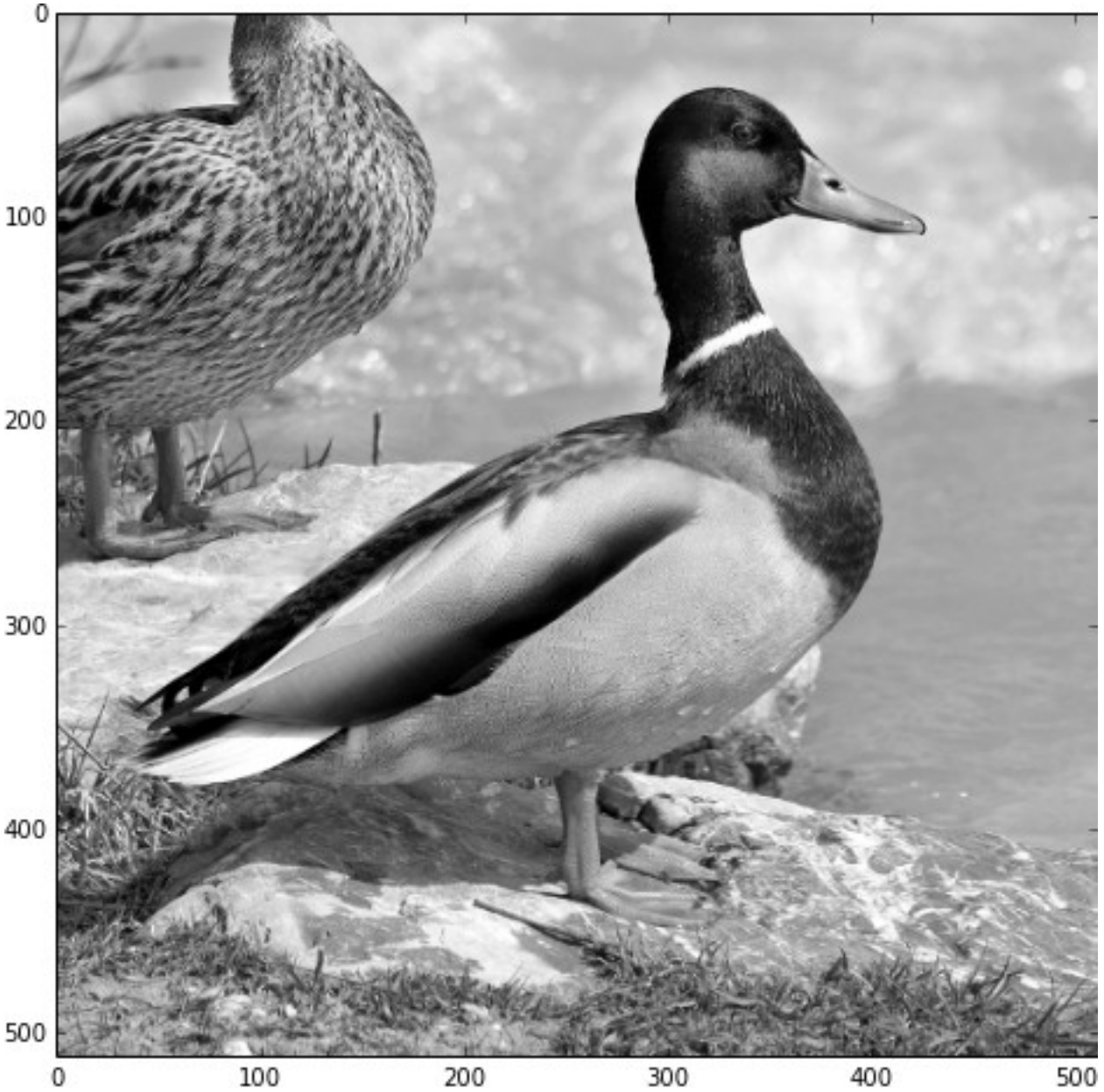
DFT Computation Time:

$$O(N_l N_m N_u N_v) \sim O(N^4)$$

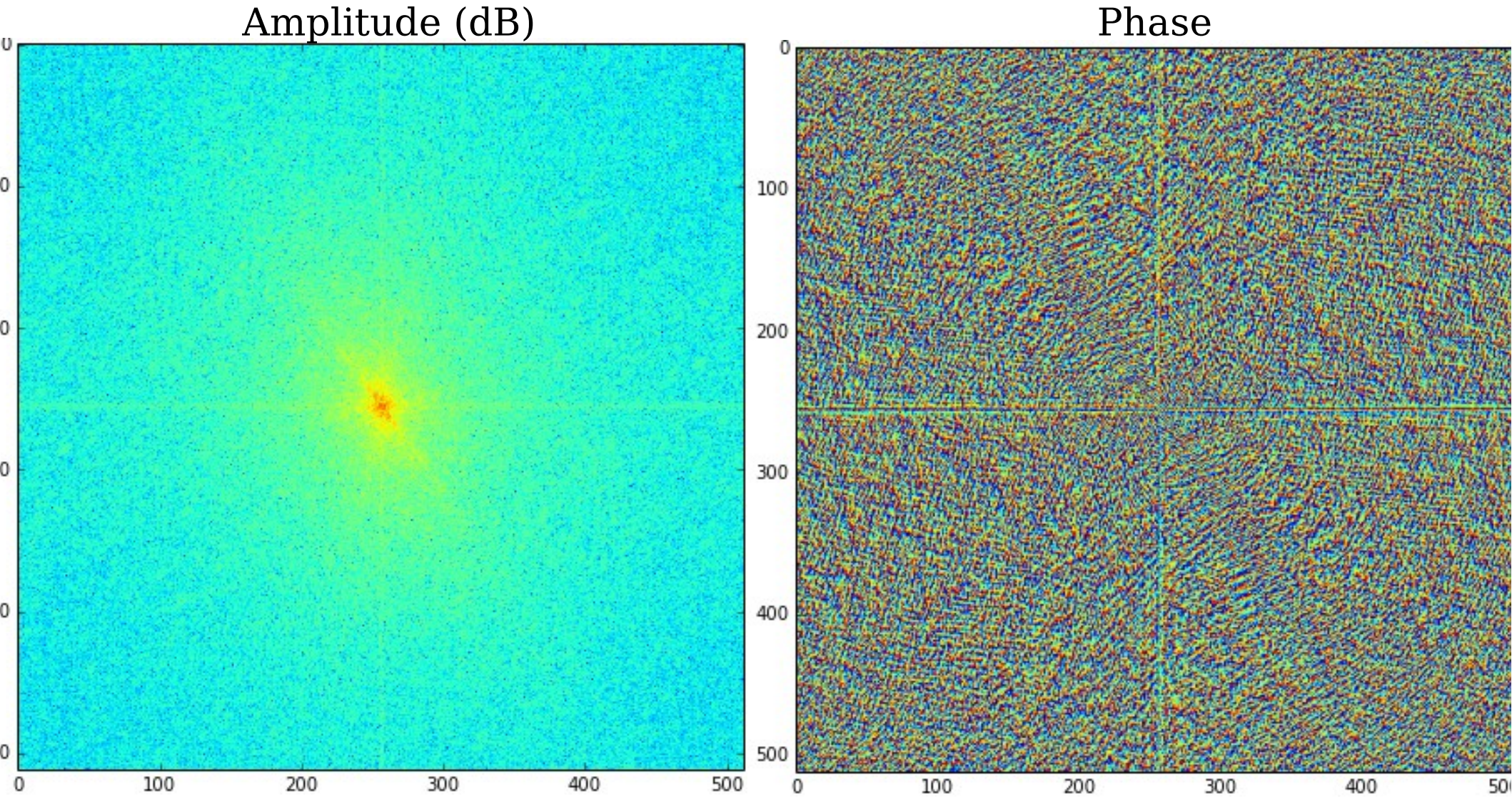
FFT's are a class of algorithms (Cooley – Tukey is a classic) which implement a DFT on periodically sampled (gridded) data which are of order

$$O(N^2 \log_2 N)$$

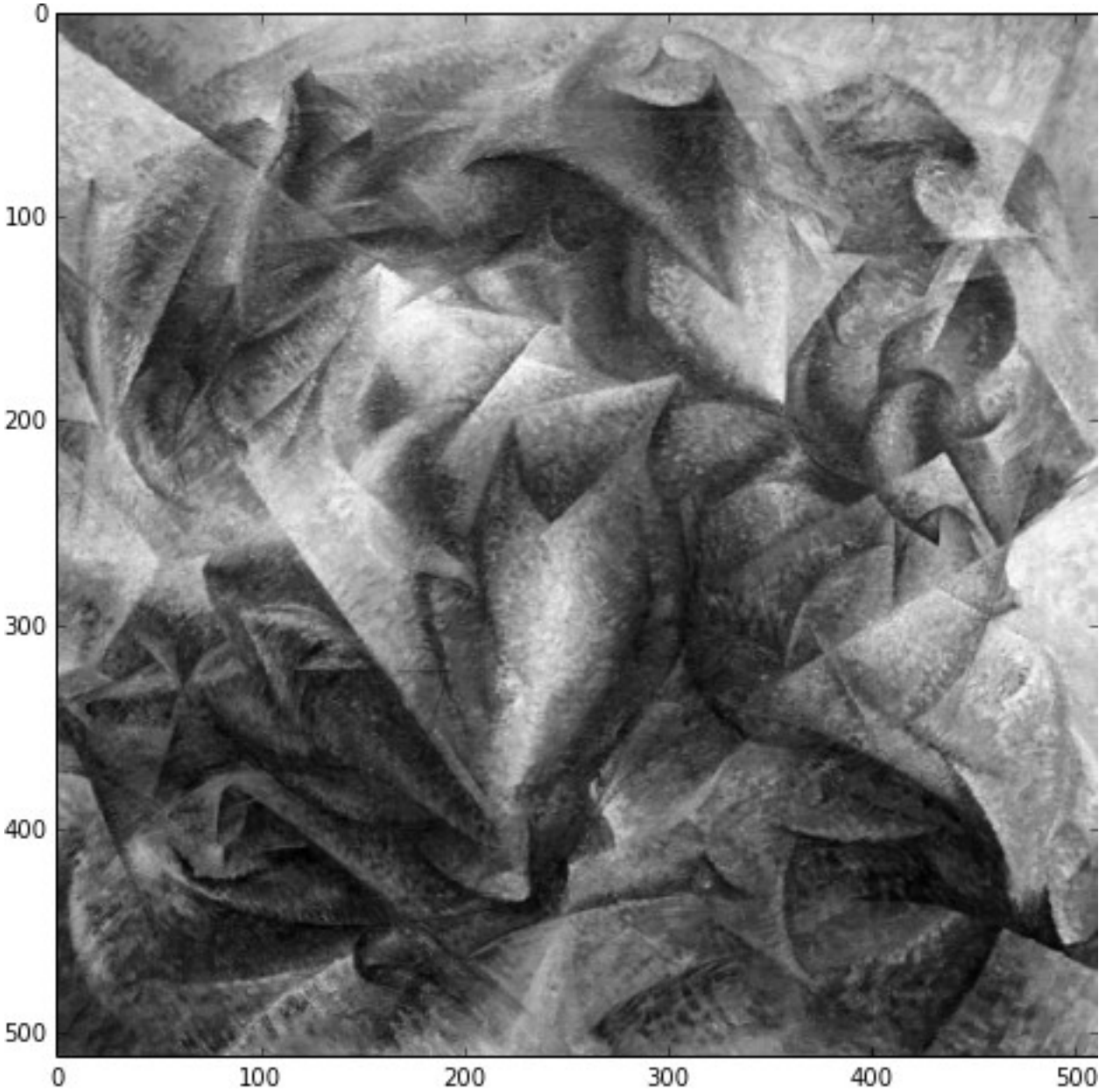
A vastly reduced computation time, this makes doing large Fourier Transforms in reasonable timescales



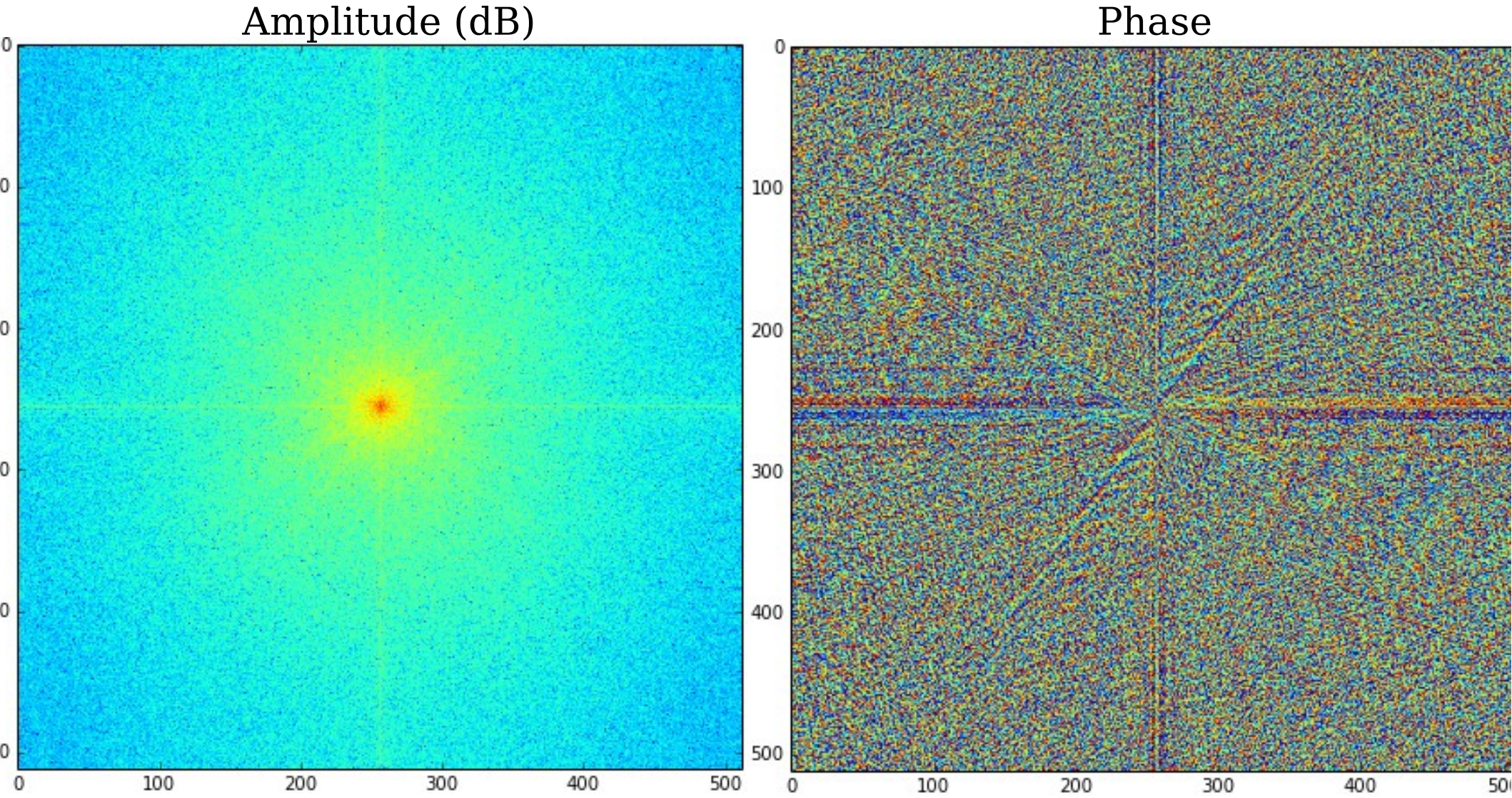
A Duck



Fourier Transform(A Duck)

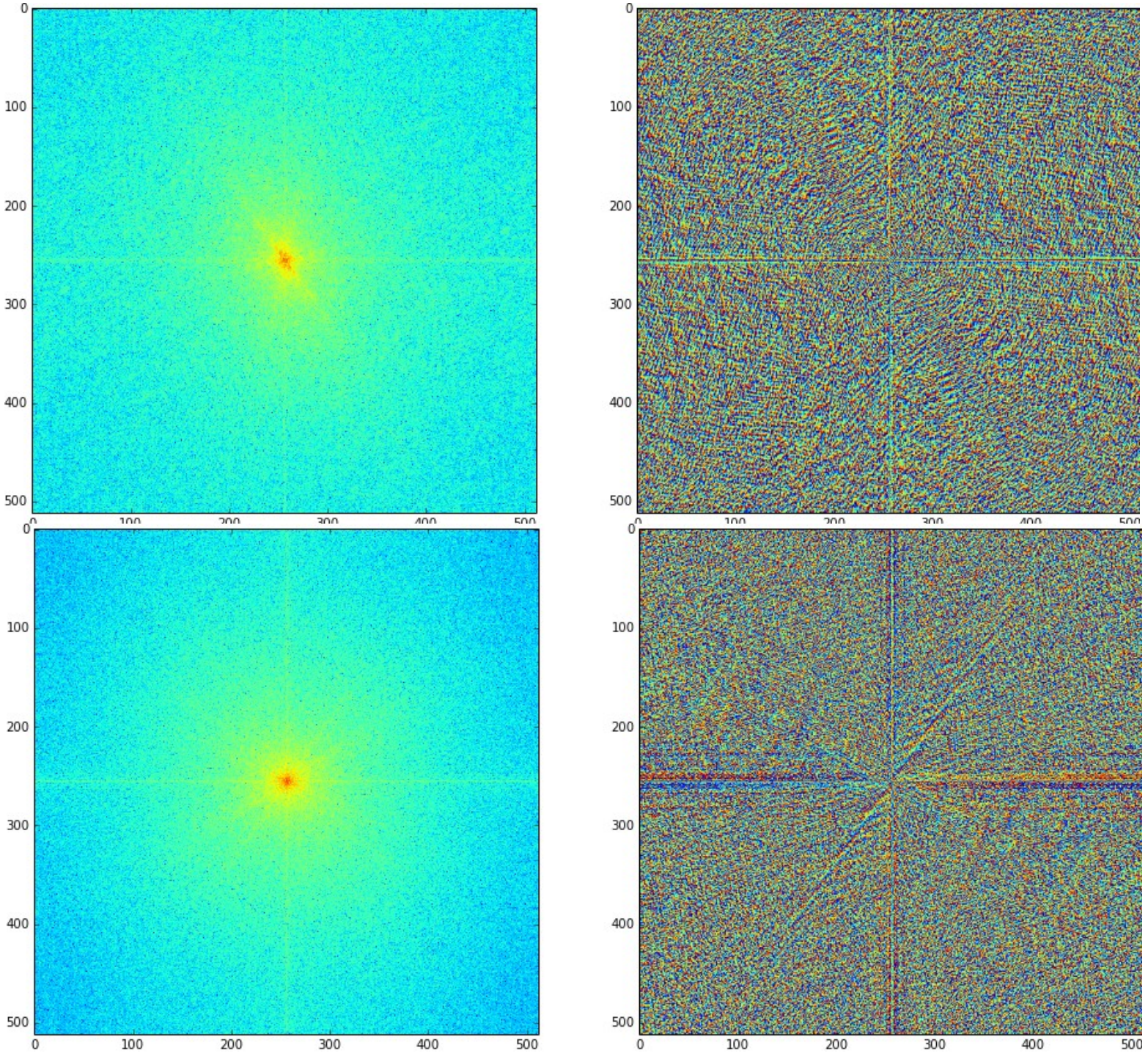


Umberto Boccioni's *Dynamism of a Soccer Player*

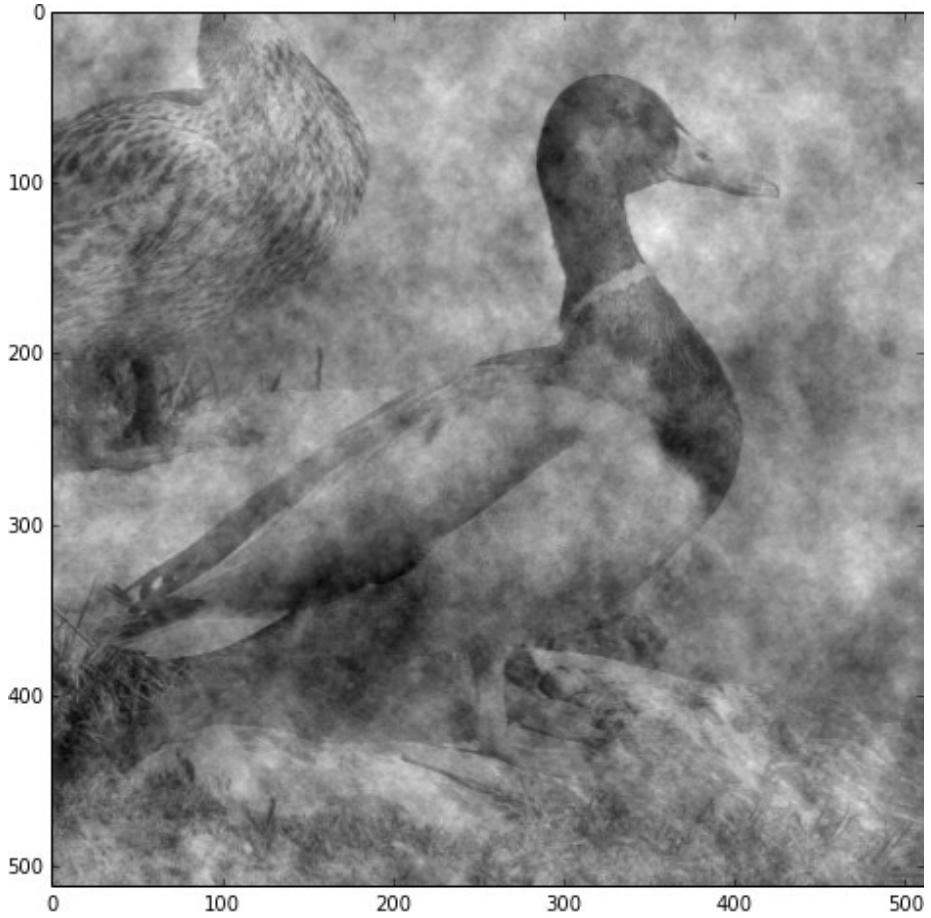


Fourier Transform(*Dynamism of a Soccer Player*)

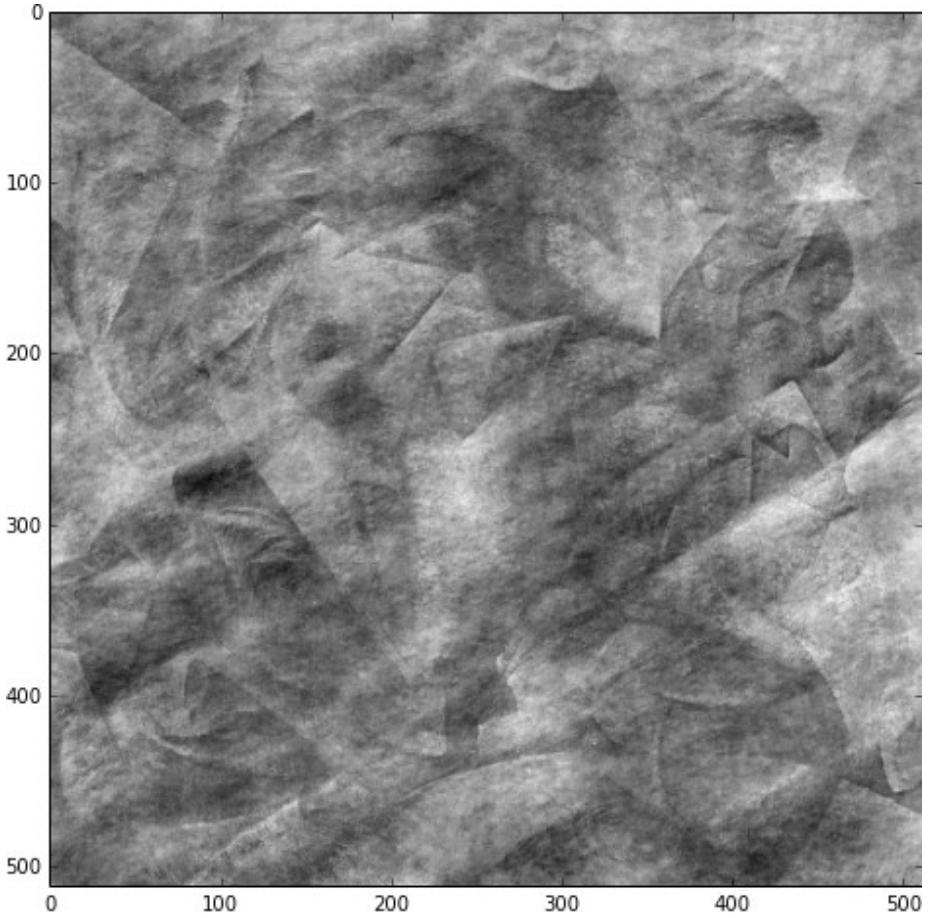
Which is a duck, and which a masterpiece of the Italian Futurism movement?



Phase and Amplitude

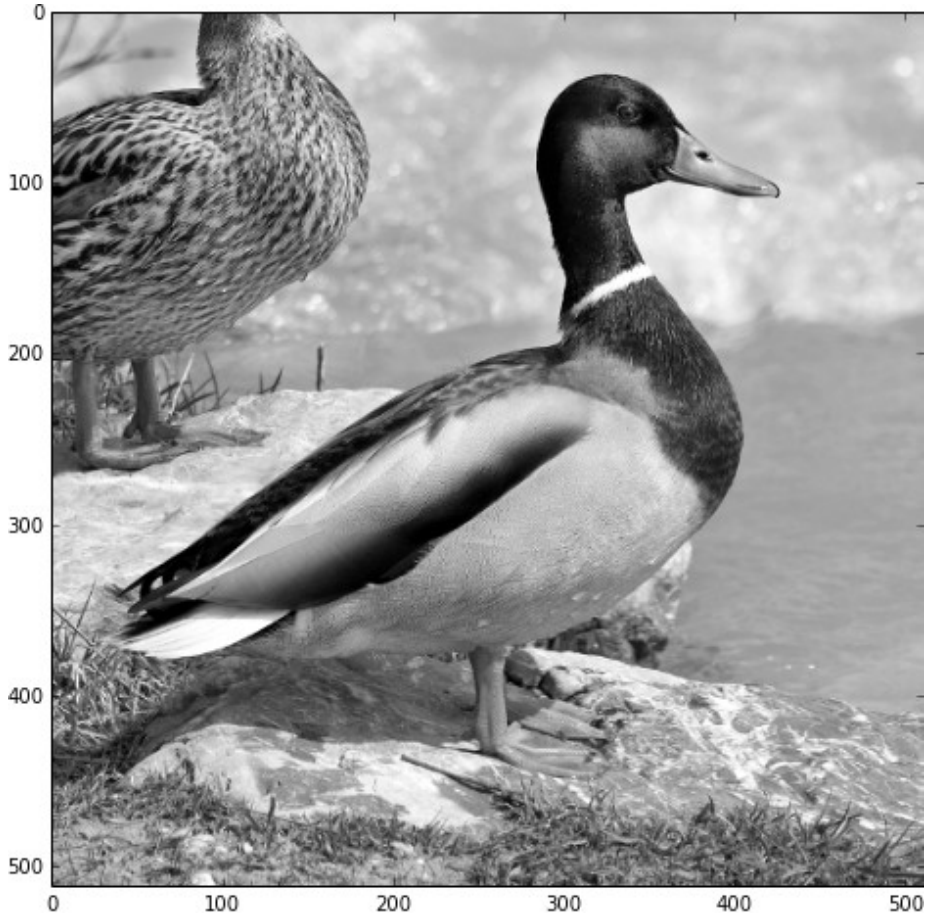


Hybrid image:
phase of a duck,
amplitude of the soccer player

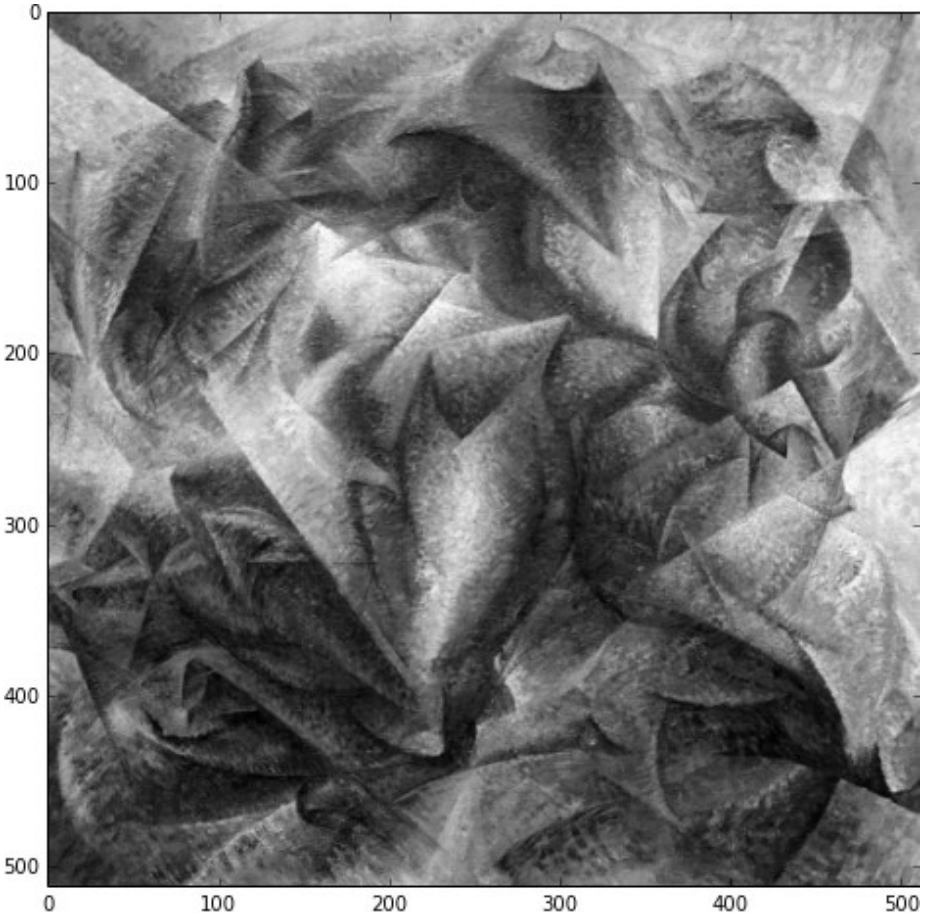


Hybrid image:
phase of the soccer player,
amplitude of a duck

Phase and Amplitude

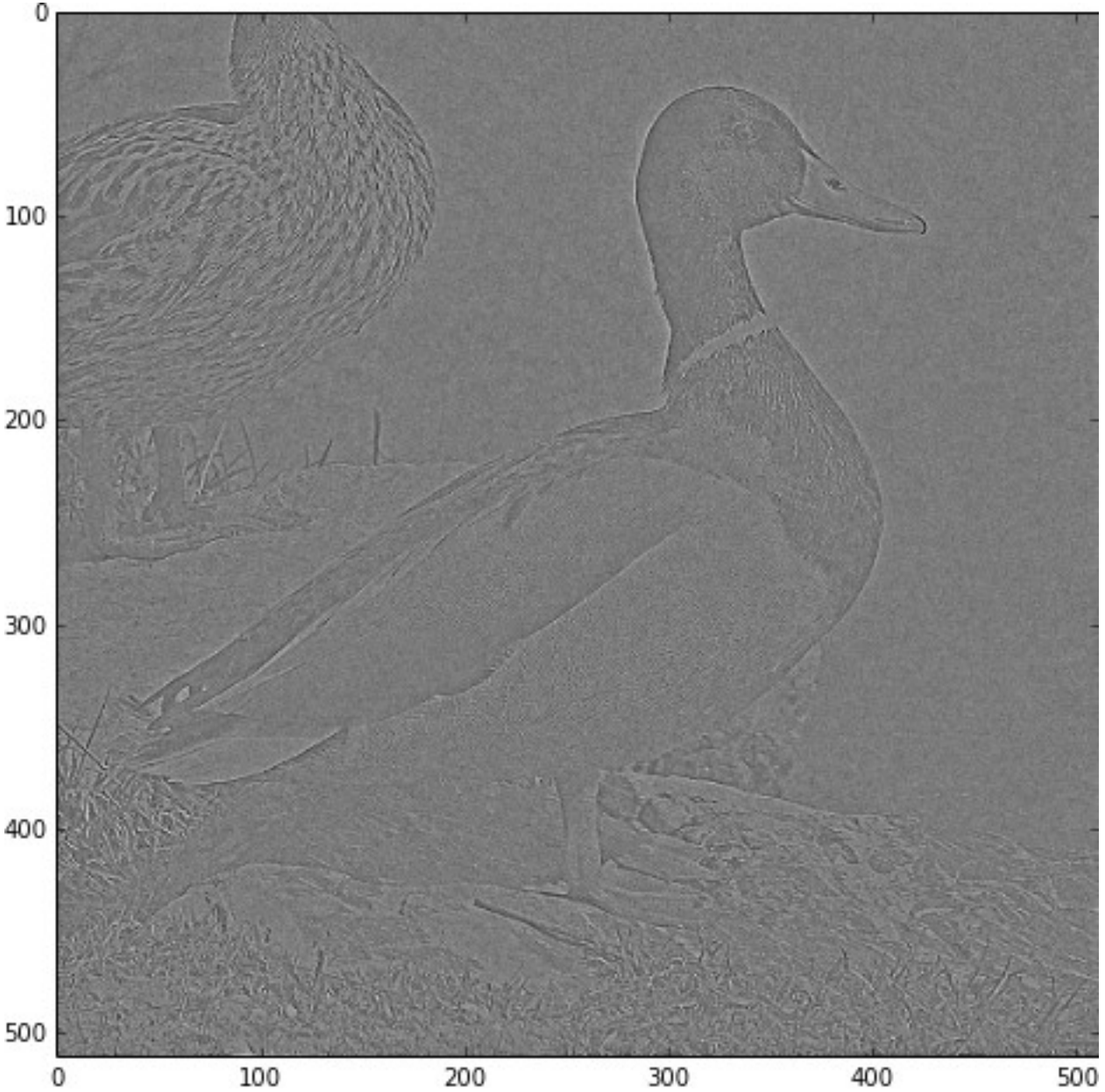


Original



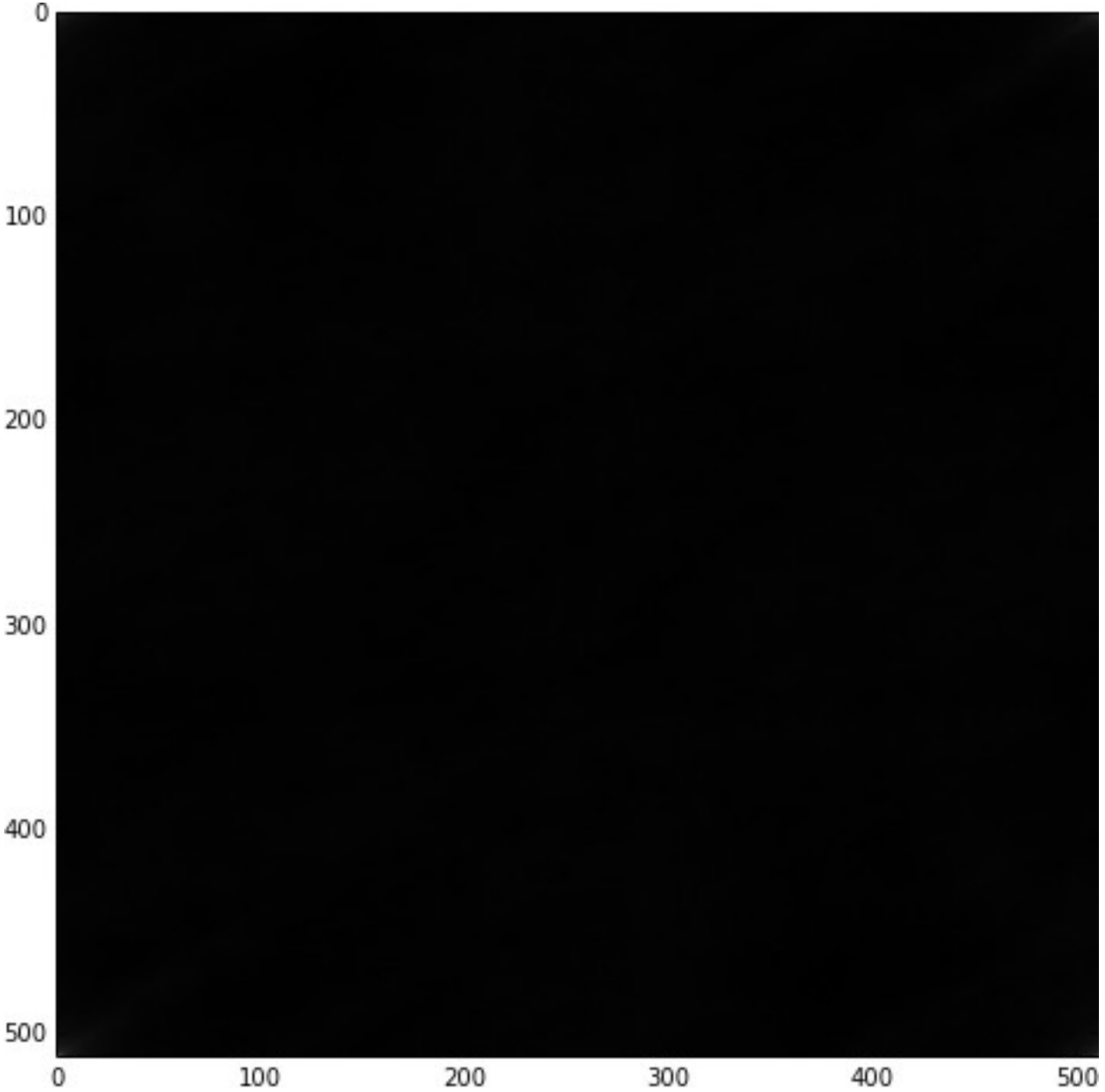
Original

High frequency components like edges are prominent, while low spatial modes are down weighted



A Duck reconstructed with only phase information

Majority of power is in the DC and low spatial mode bins → the image is effectively a constant field without phase information



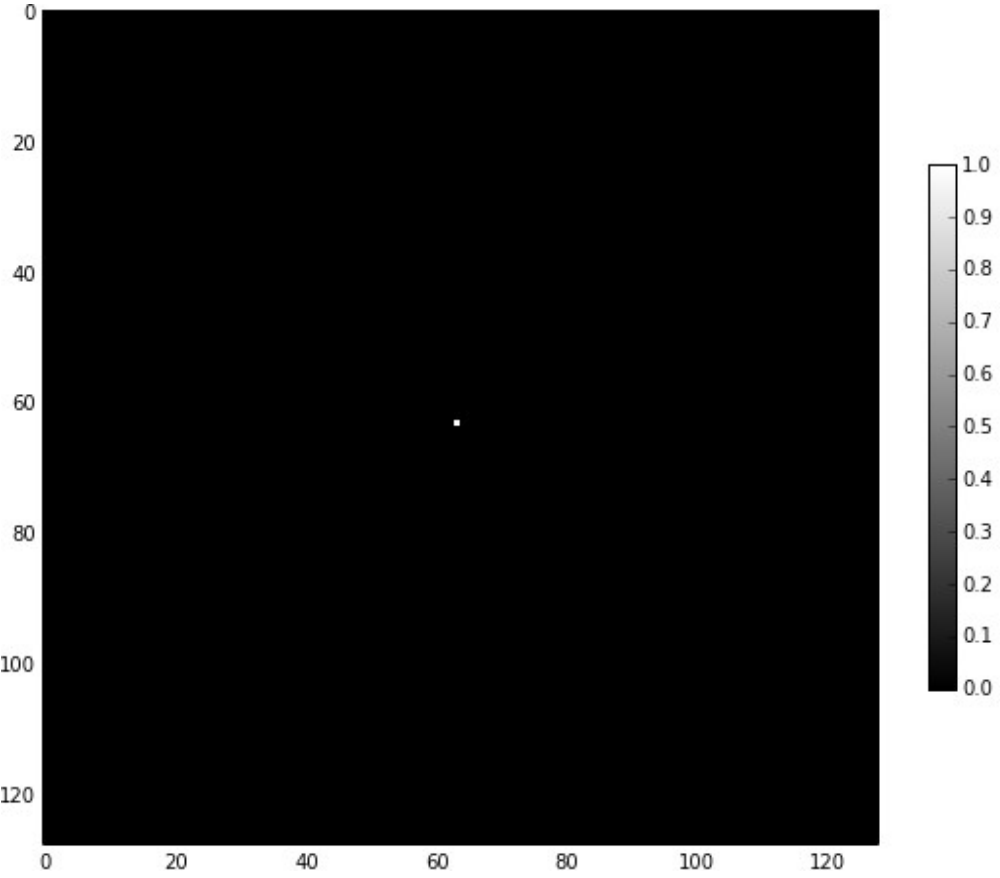
A Duck reconstructed with only amplitude information

Amplitude (flux) depends on weightings and tapers, which emphasises particular spatial modes over others, but without phase information we have no idea how that flux is distributed.

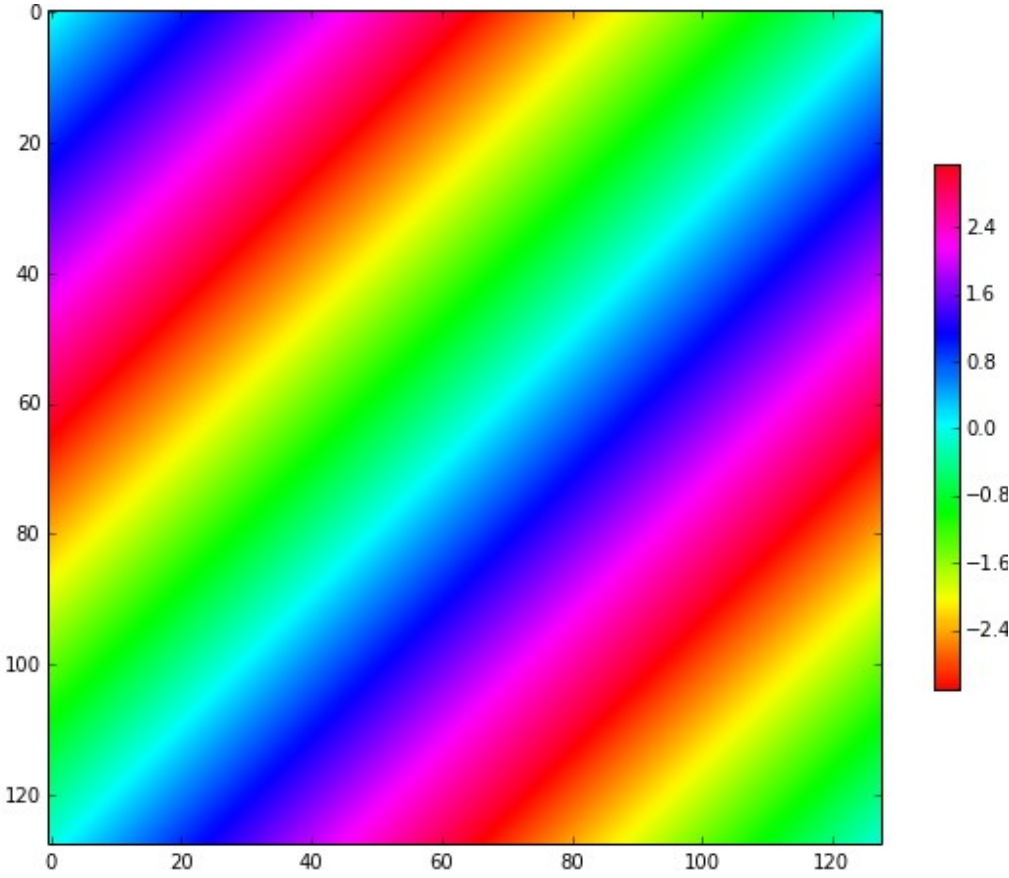
We will see later in the talk how and why we would tune the amplitude.

The visibilities from simple skies, what an interferometer potentially measures

Simple Sky: point source (delta function) sky

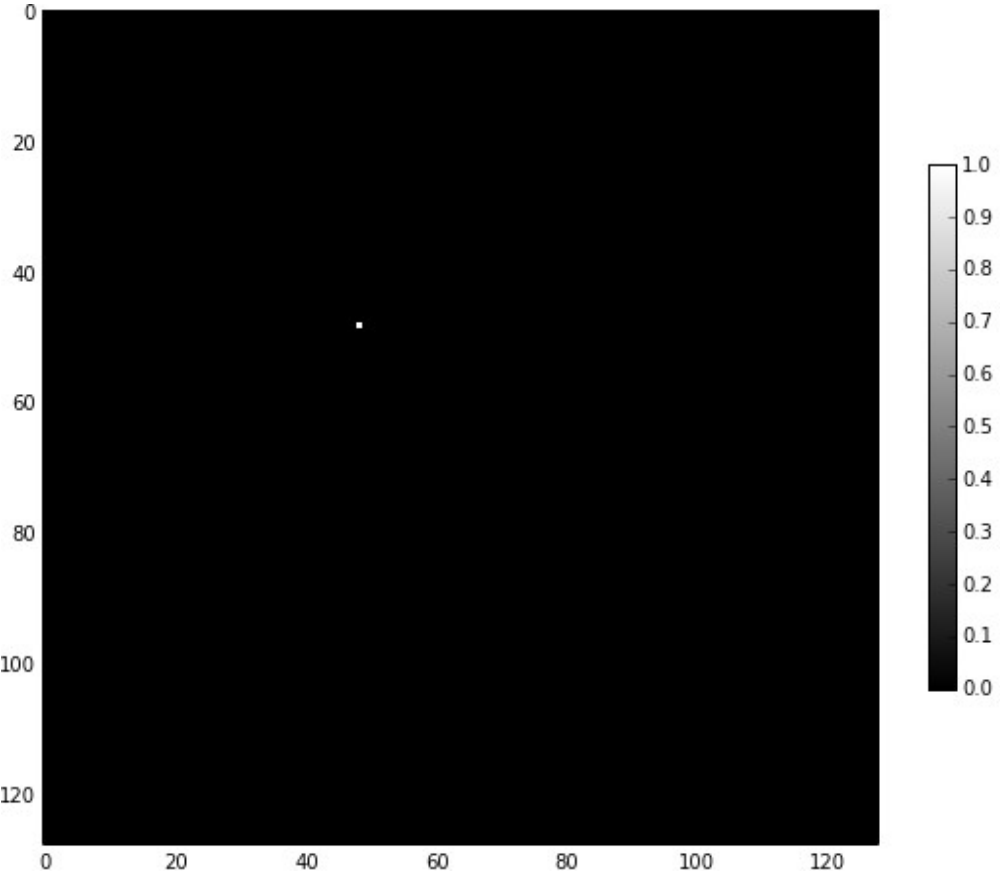


Sky



FT(Sky).phase

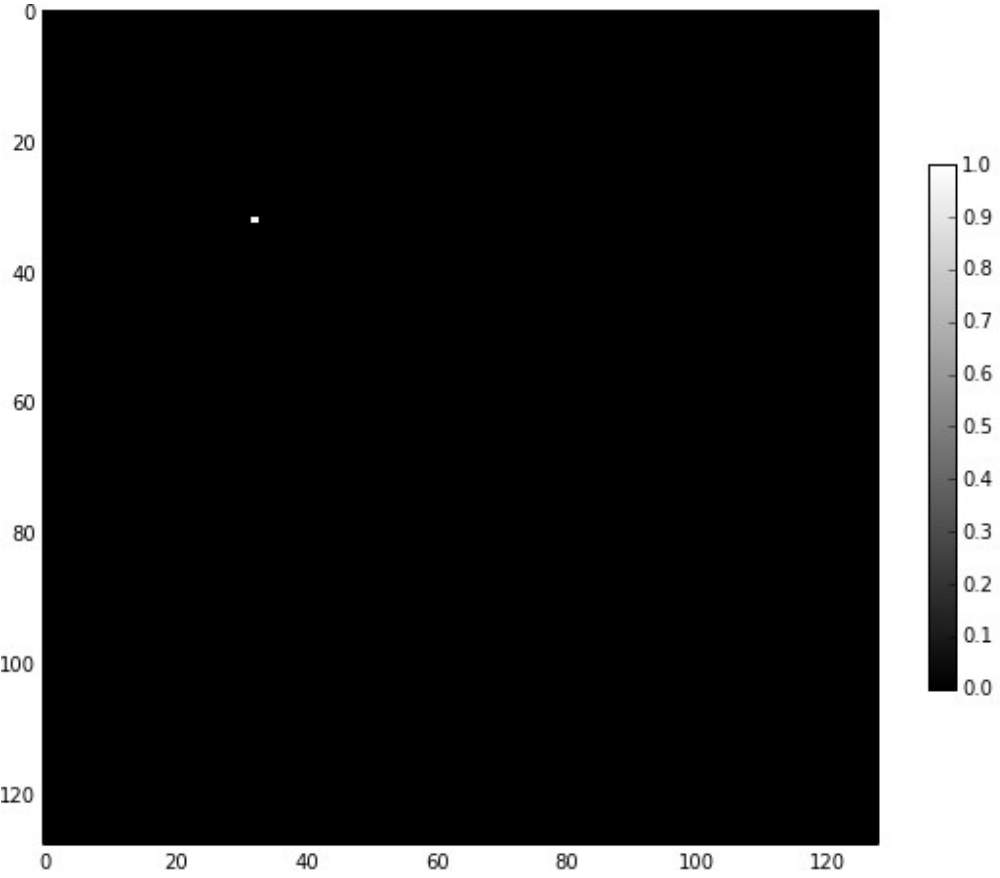
Simple Sky: point source (delta function) sky



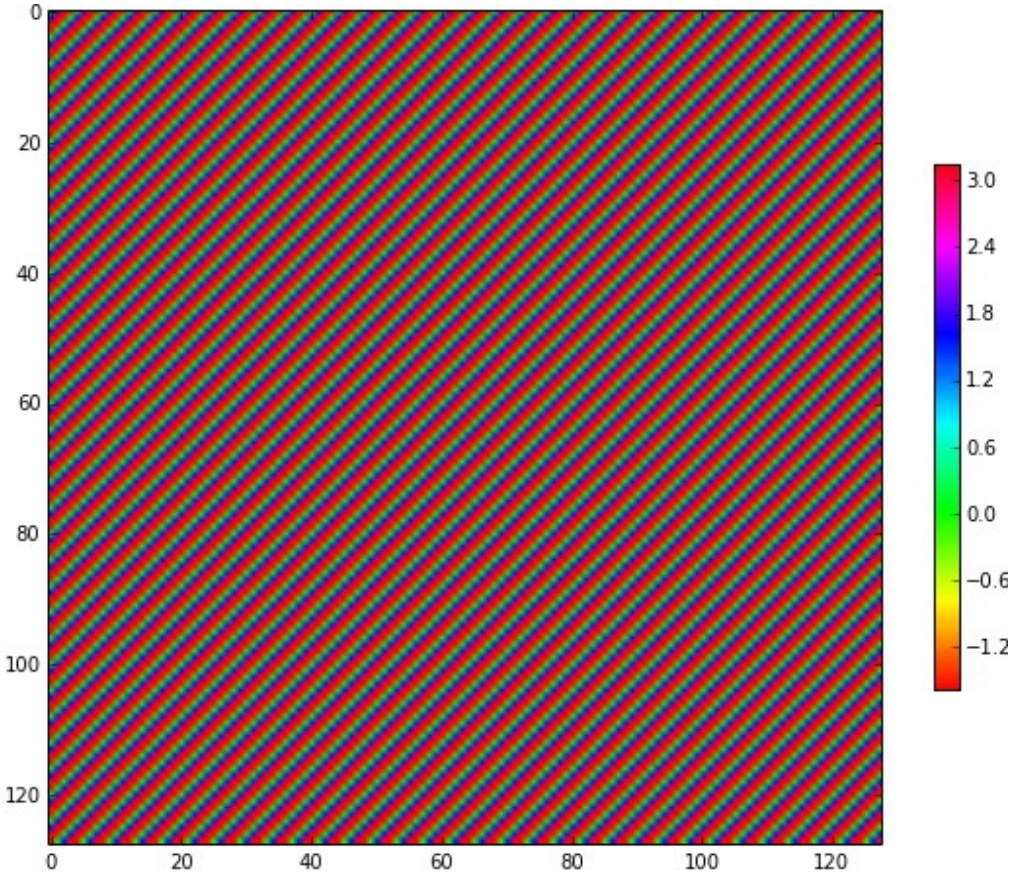
Sky

FT(Sky).phase

Simple Sky: point source (delta function) sky

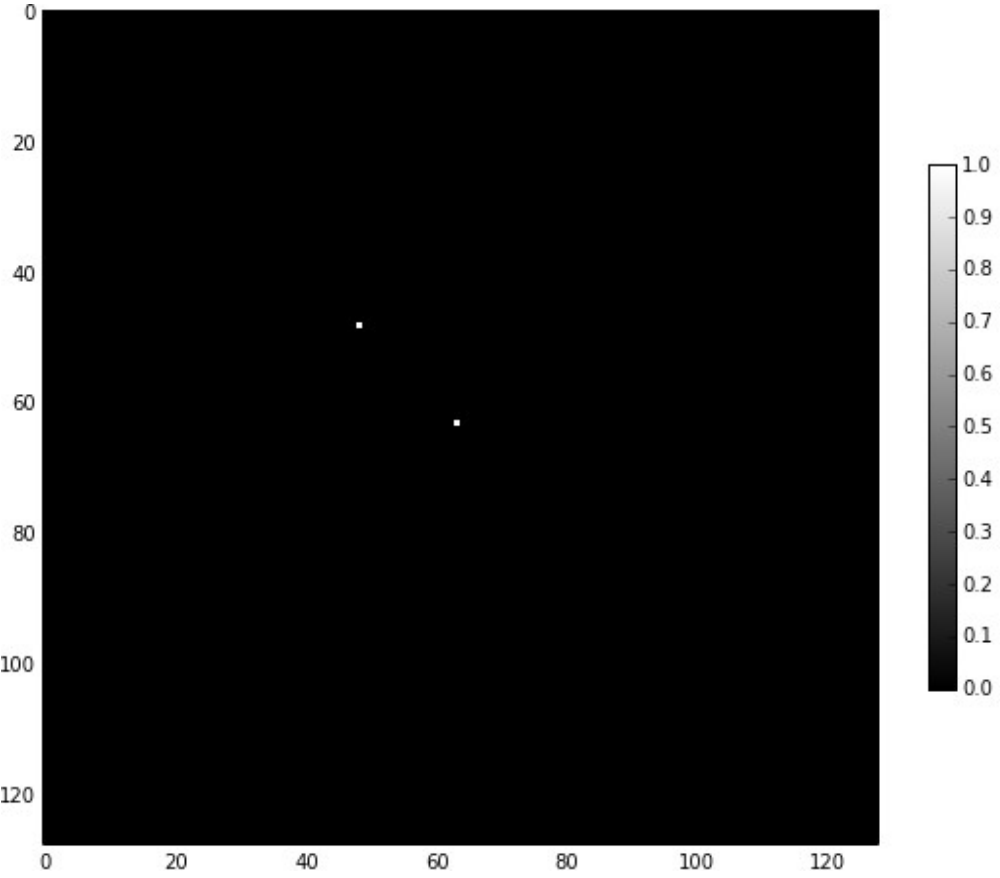


Sky

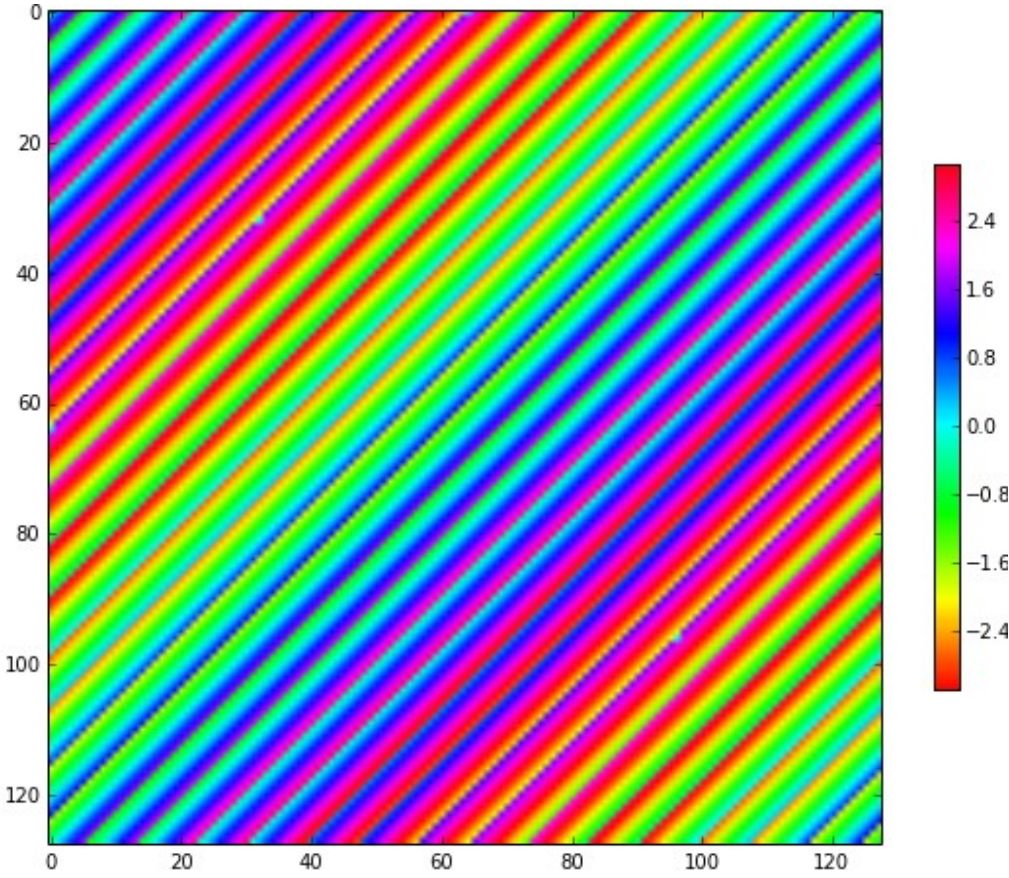


FT(Sky).phase

Simple Sky: two point source (delta function) sky

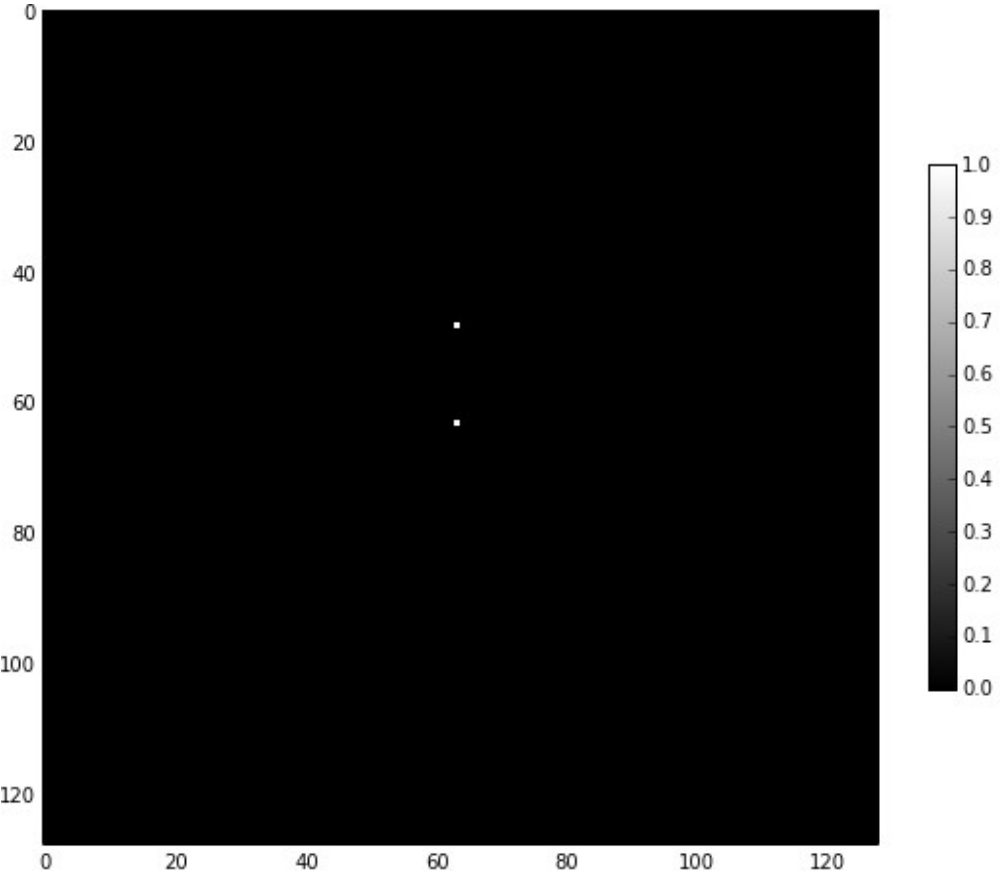


Sky

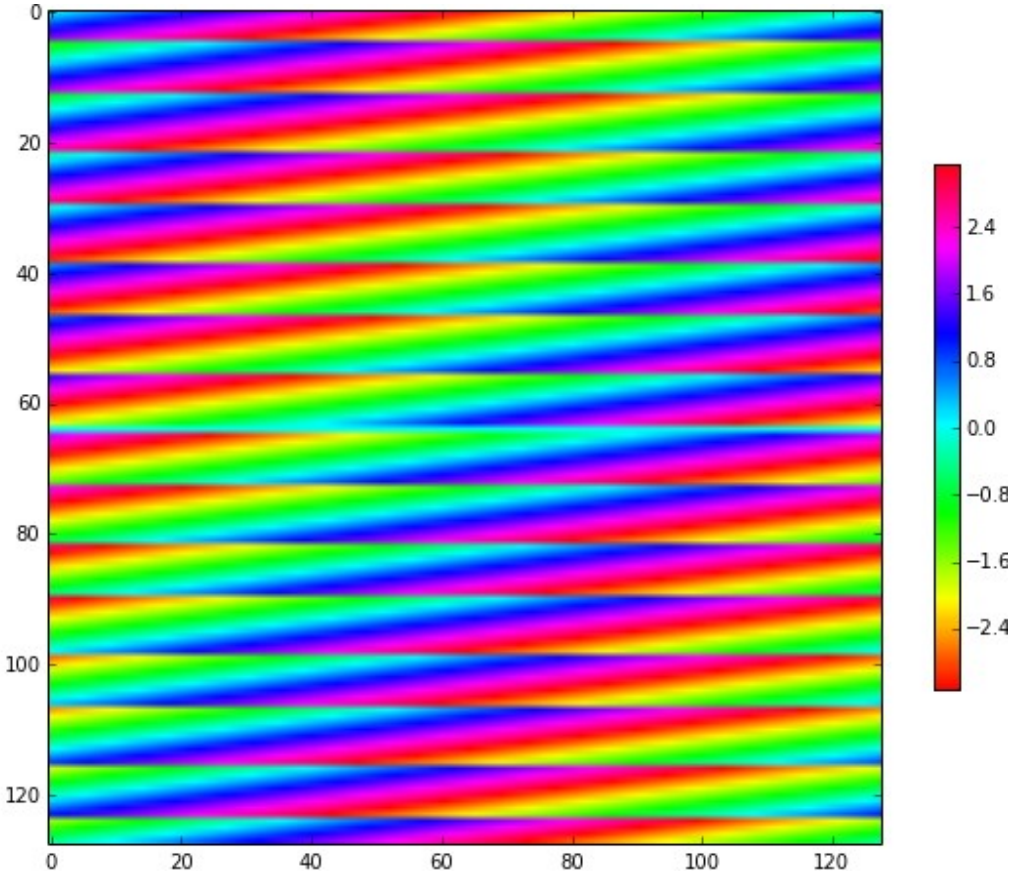


FT(Sky).phase

Simple Sky: two point source (delta function) sky

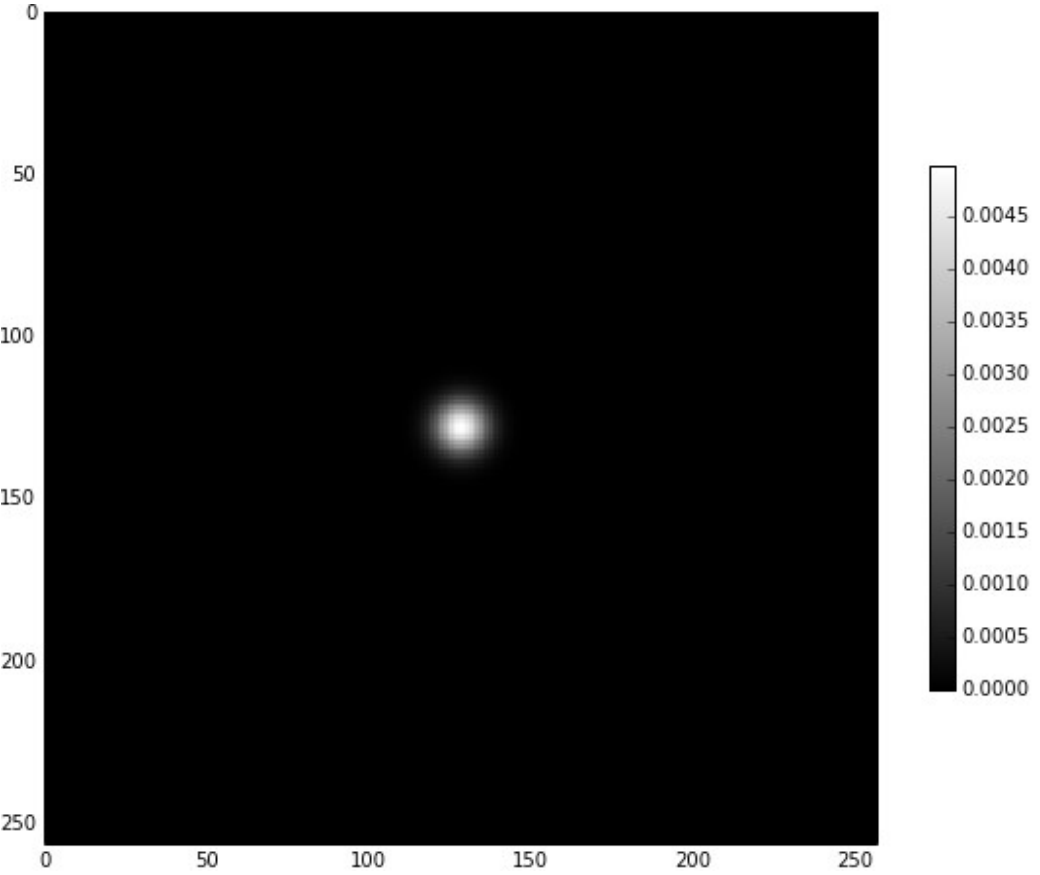


Sky

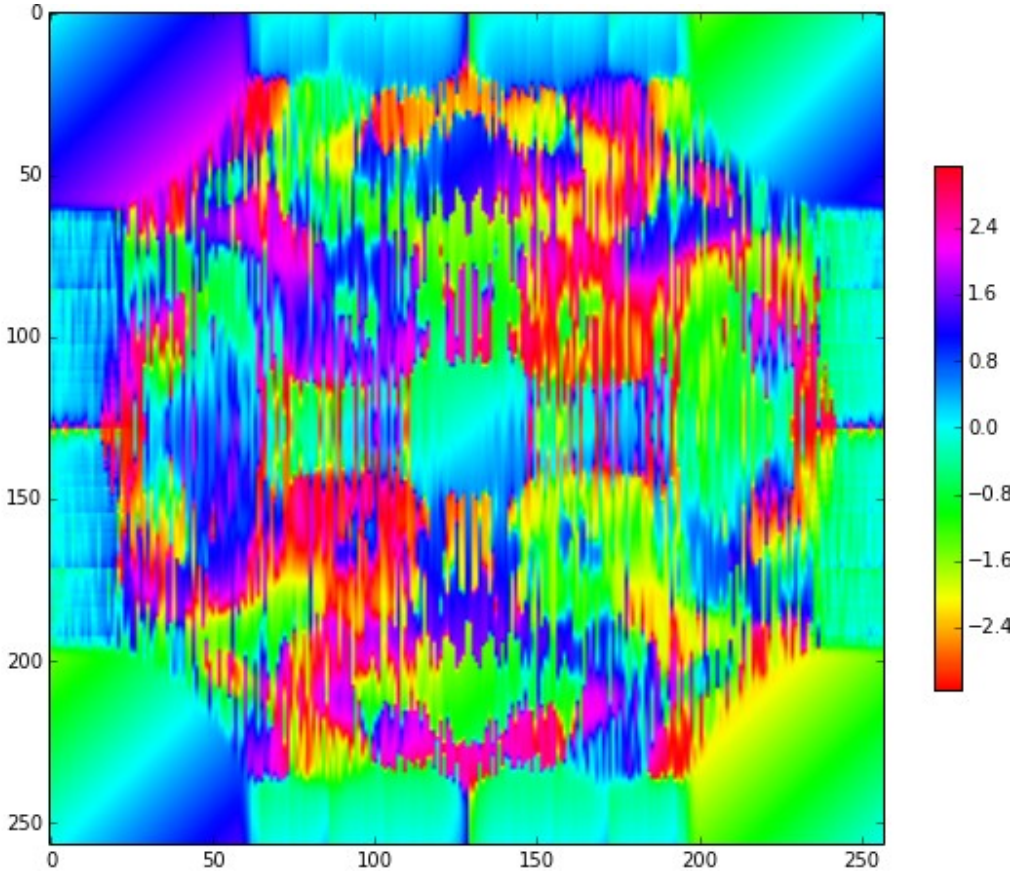


FT(Sky).phase

Simple Sky: Diffuse source (Gaussian)

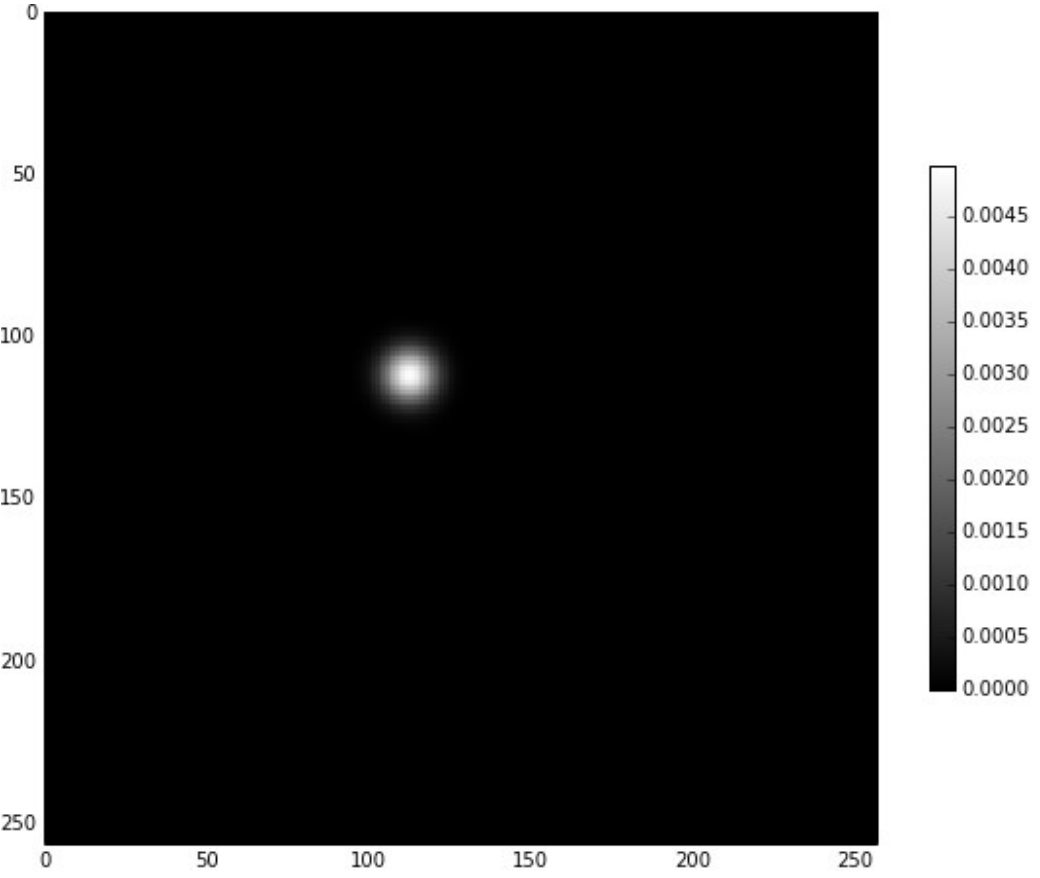


Sky

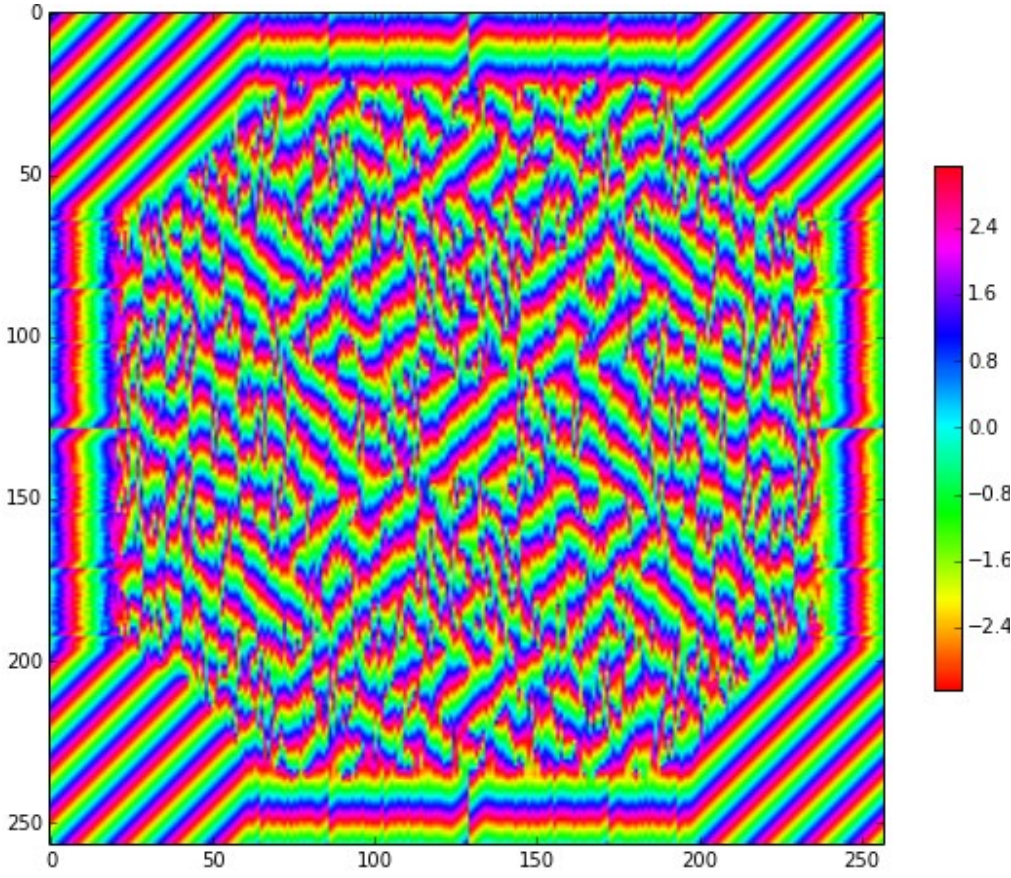


FT(Sky).phase

Simple Sky: Diffuse source (Gaussian)

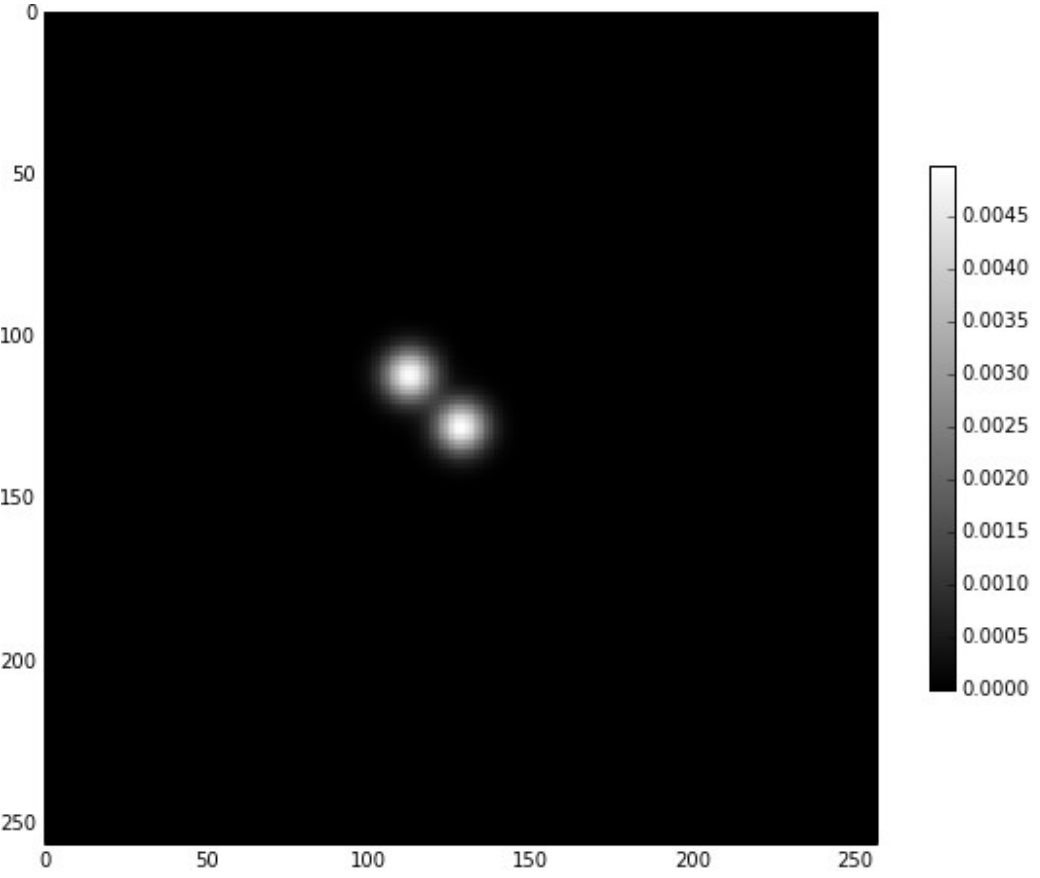


Sky

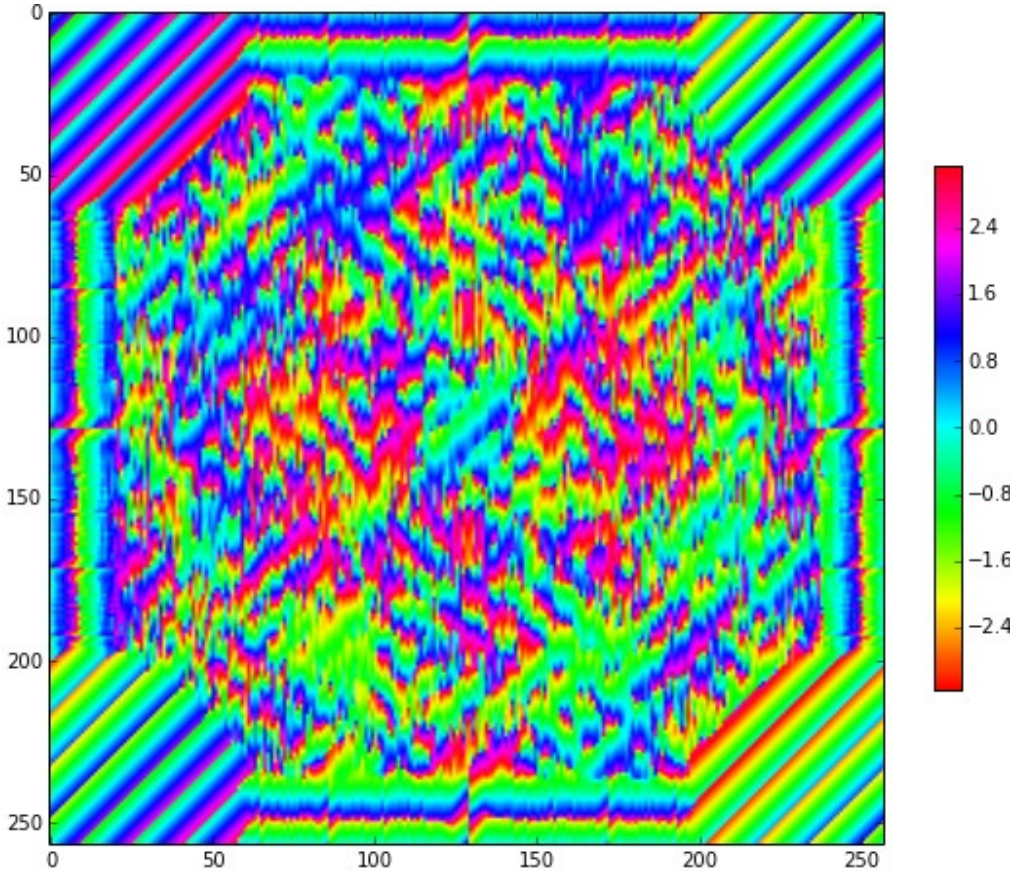


FT(Sky).phase

Simple Sky: Diffuse source (Gaussian)

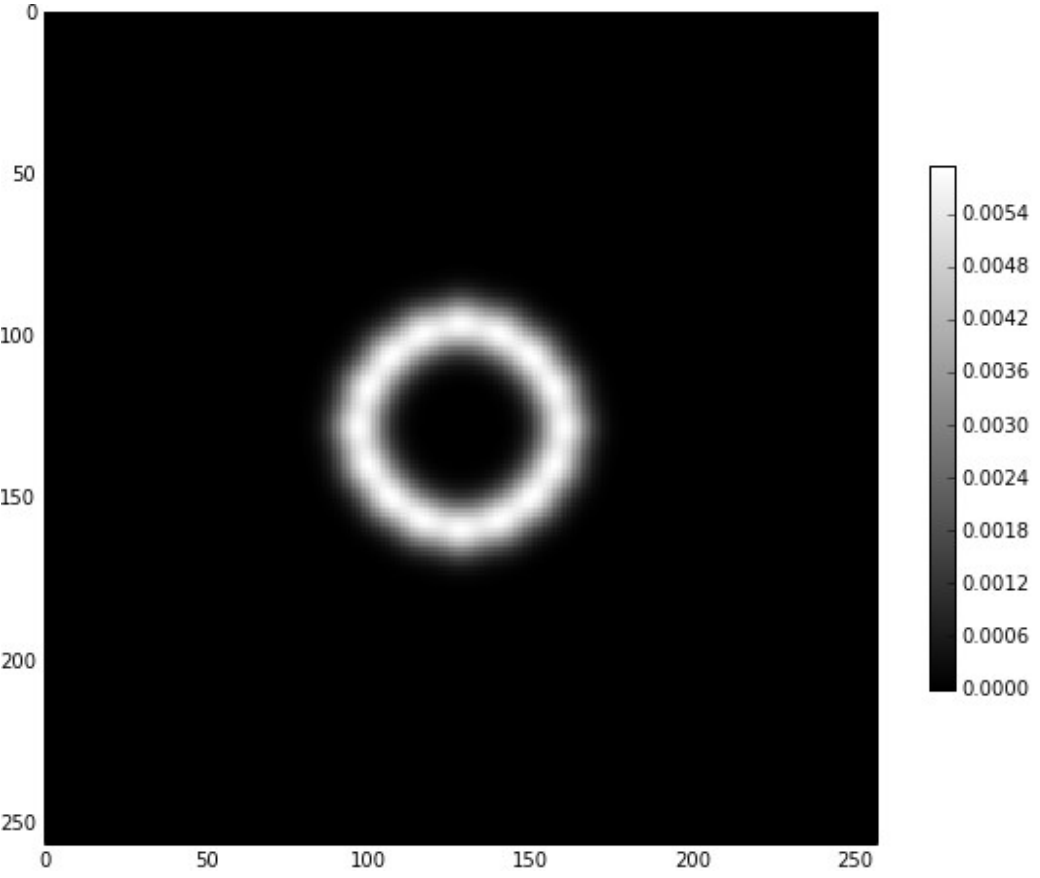


Sky

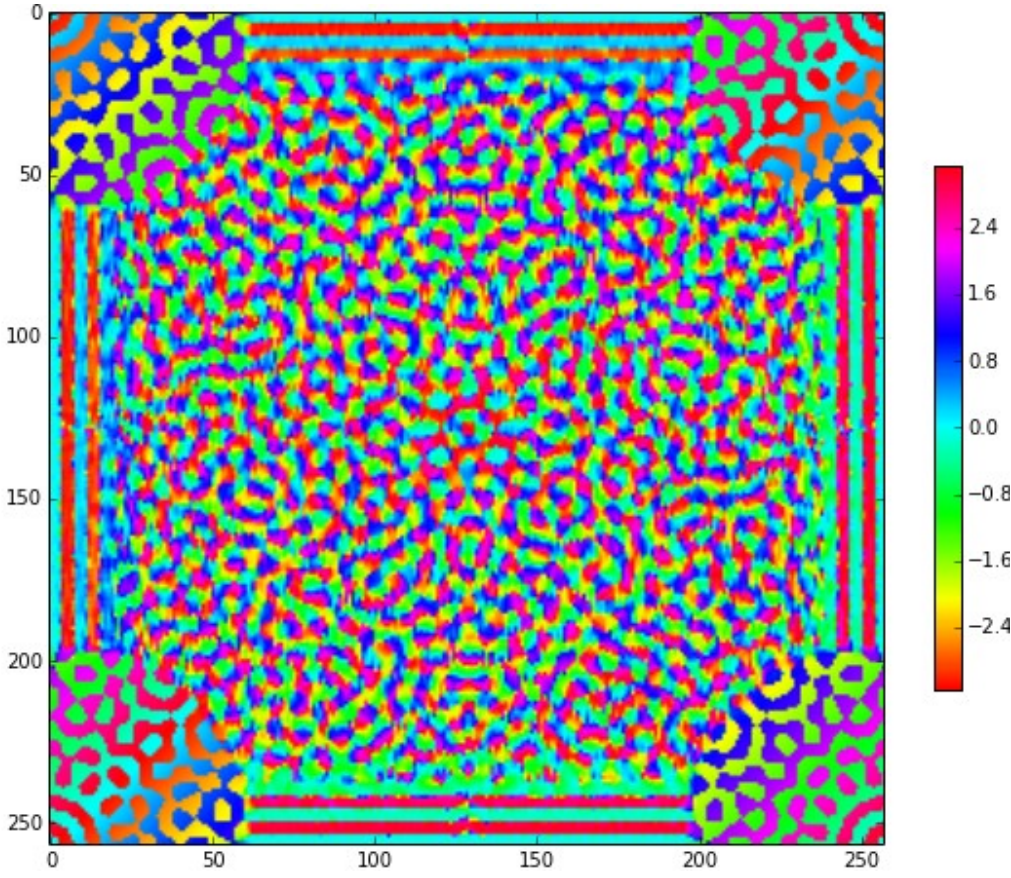


FT(Sky).phase

Simple Sky: Diffuse source (Gaussian)



Sky



FT(Sky).phase

We don't measure the entire visibility space, we sample it based on the array configuration. This gives rise to the Point Spread Function (PSF) and weighting function

The true image is the Fourier transform of the true visibilities

$$I = \mathcal{F}(V_{true})$$

But, in our interferometric array we measure noisy visibilities with some distribution (S is the sampling function) in the visibility domain. Thus we get a 'dirty' image (I^D)

$$I^D = \mathcal{F}(SV_{obs}) = \mathcal{F}(S) \circ \mathcal{F}(V_{obs})$$

The Point Spread Function (PSF) is the Fourier transform of the sampling function

$$PSF = \mathcal{F}(S)$$

Sampling Function (S)

PSF (dB)

SV

Dirty Image (I^D)

Sampling Function (S)

PSF (dB)

SV

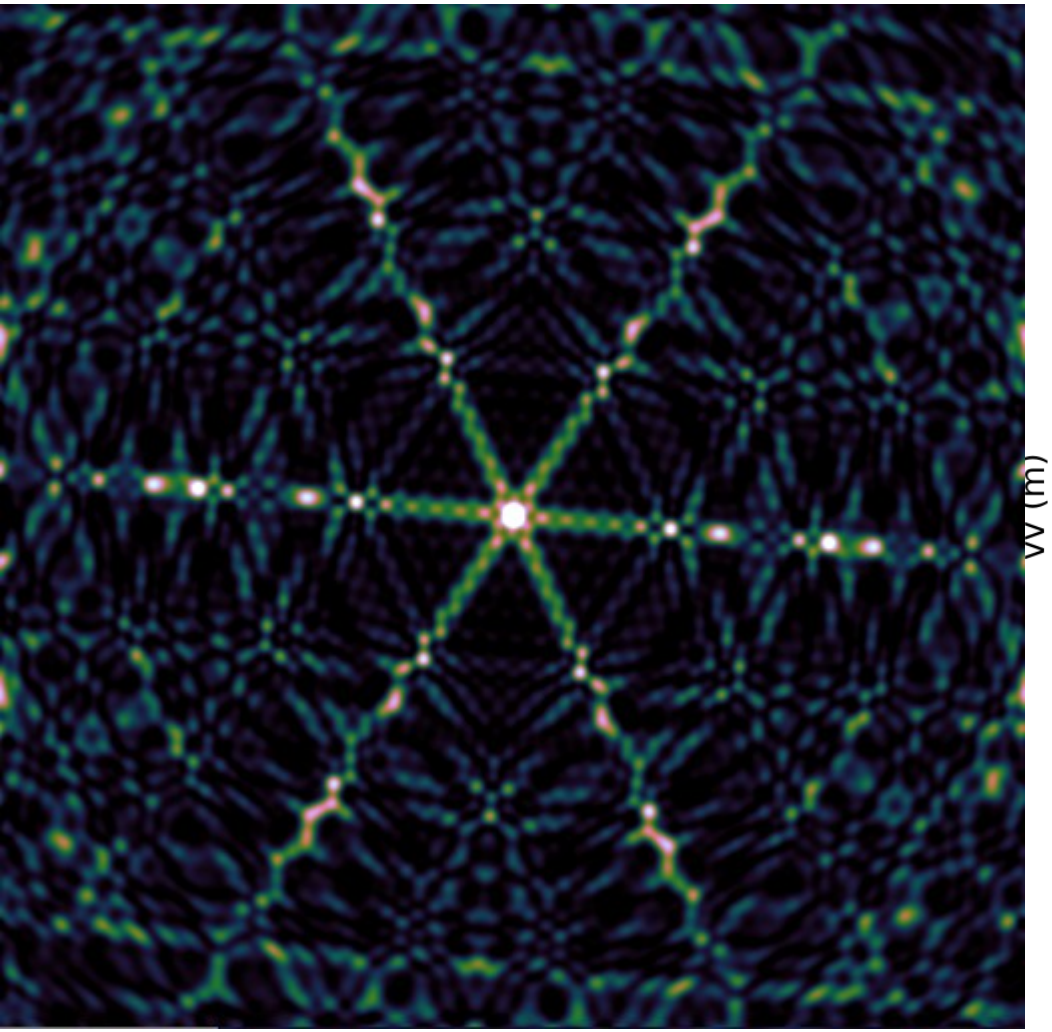
Dirty Image (I^D)

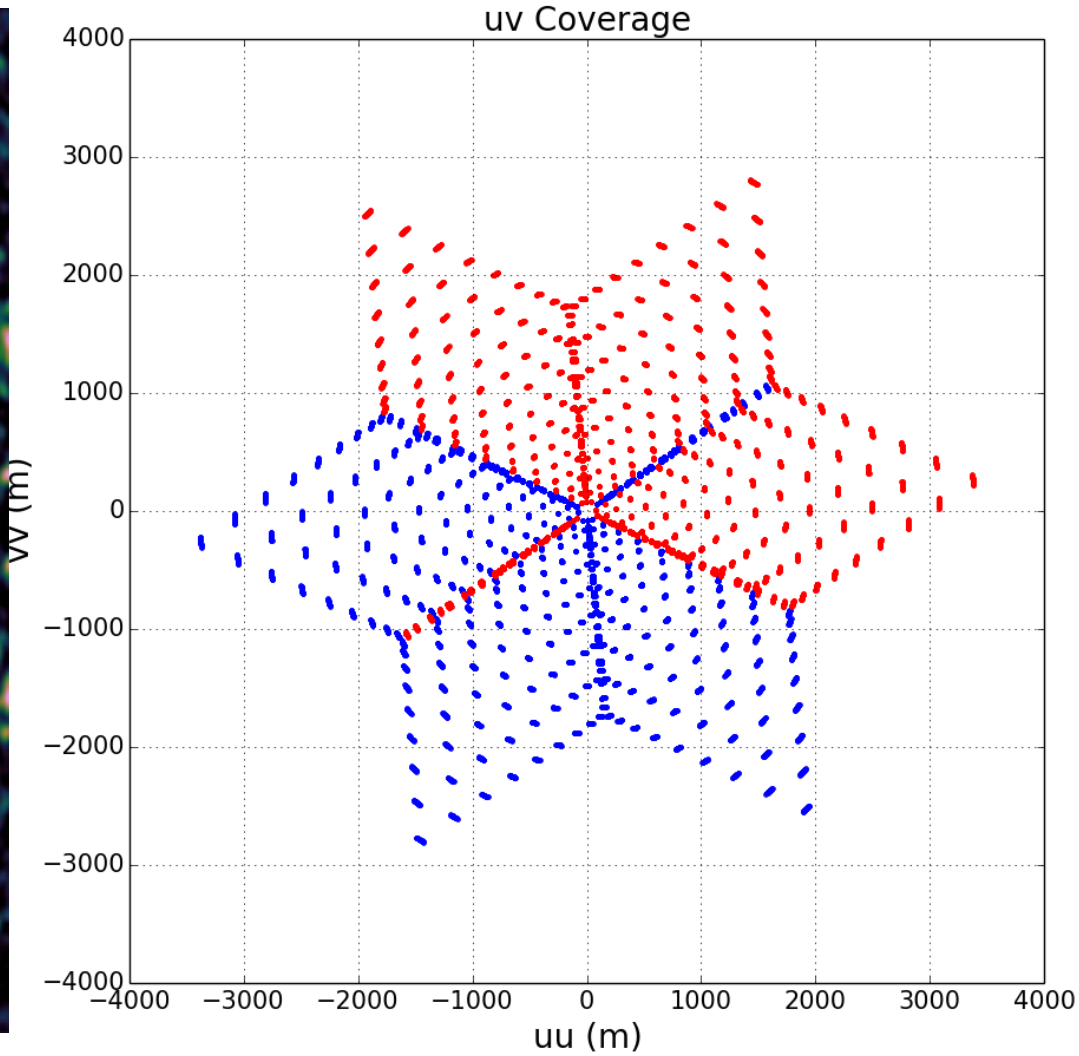
Sampling Function (S)

PSF (dB)

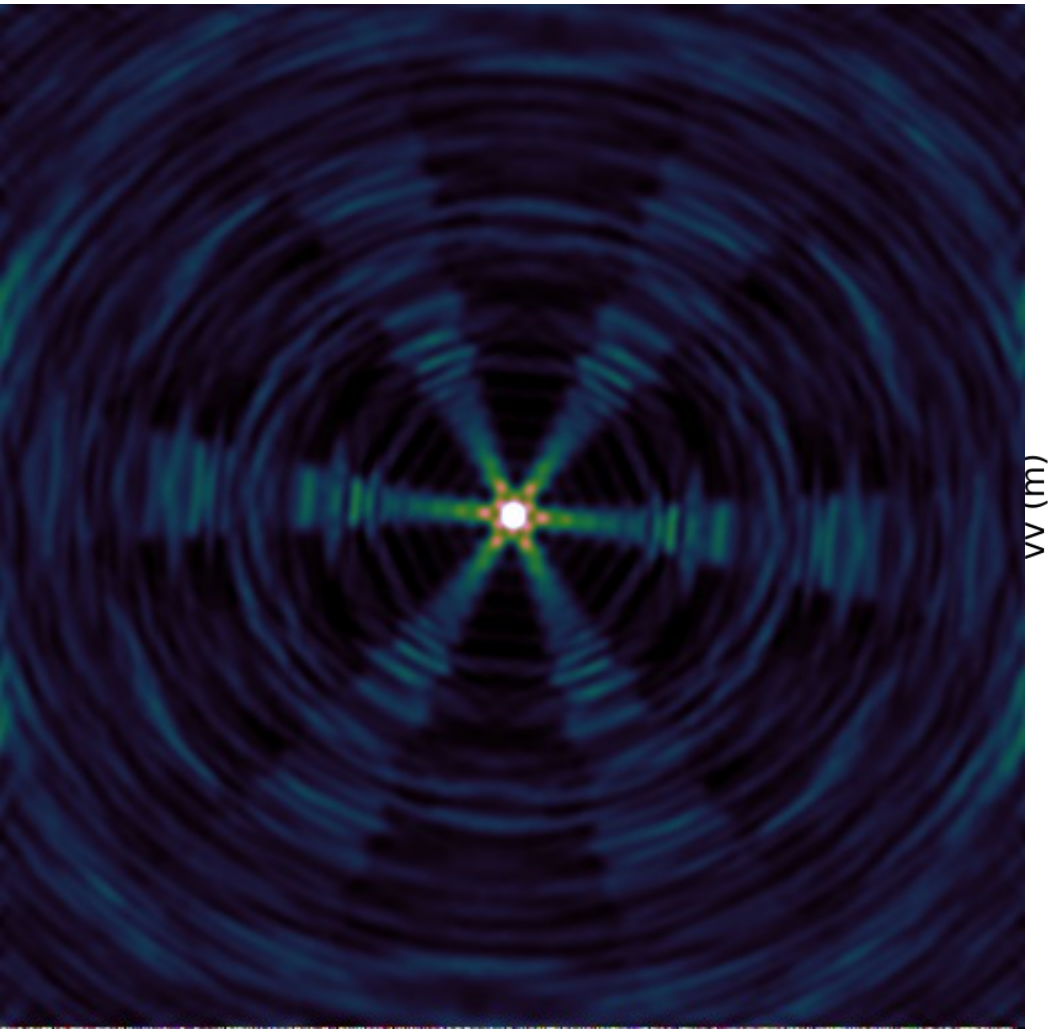
SV

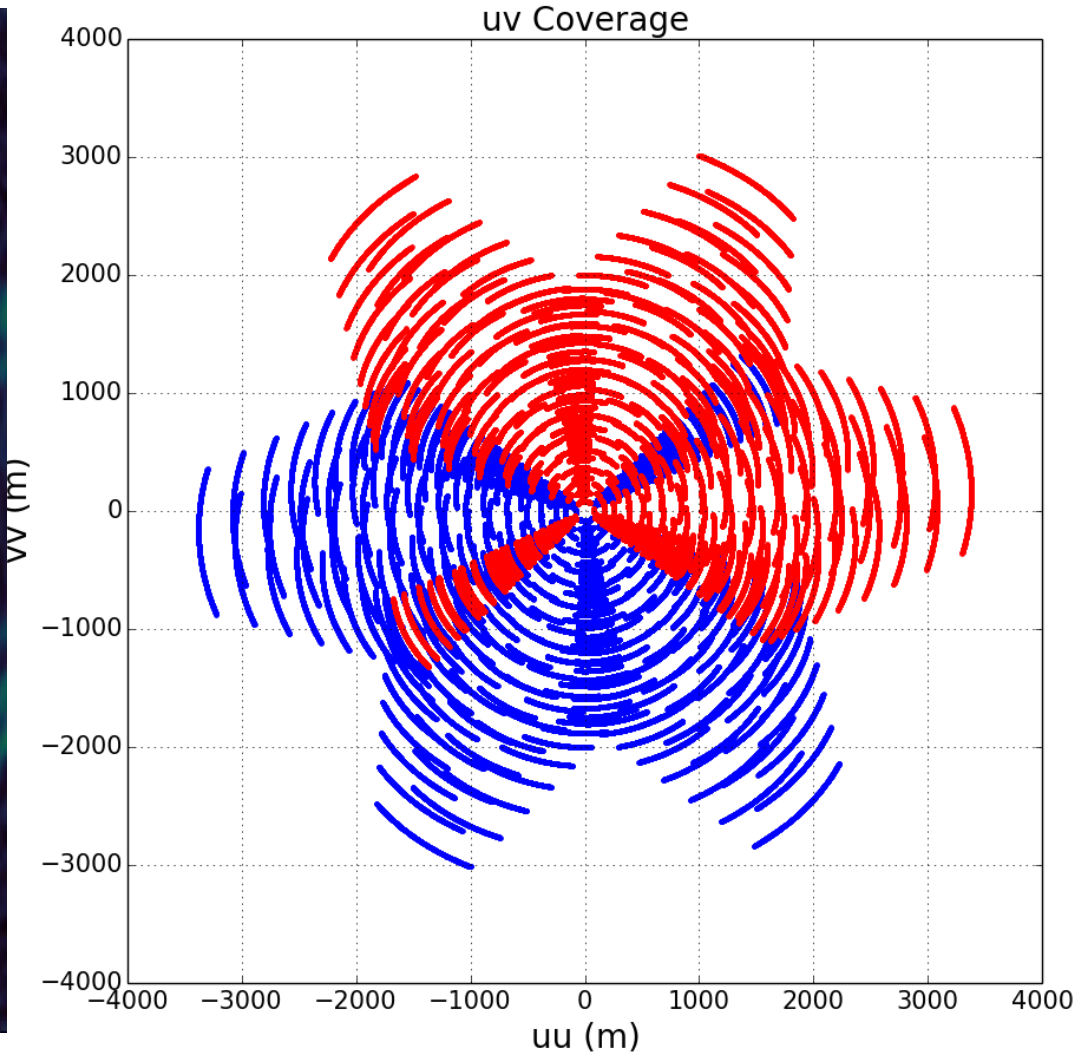
Dirty Image (I^D)



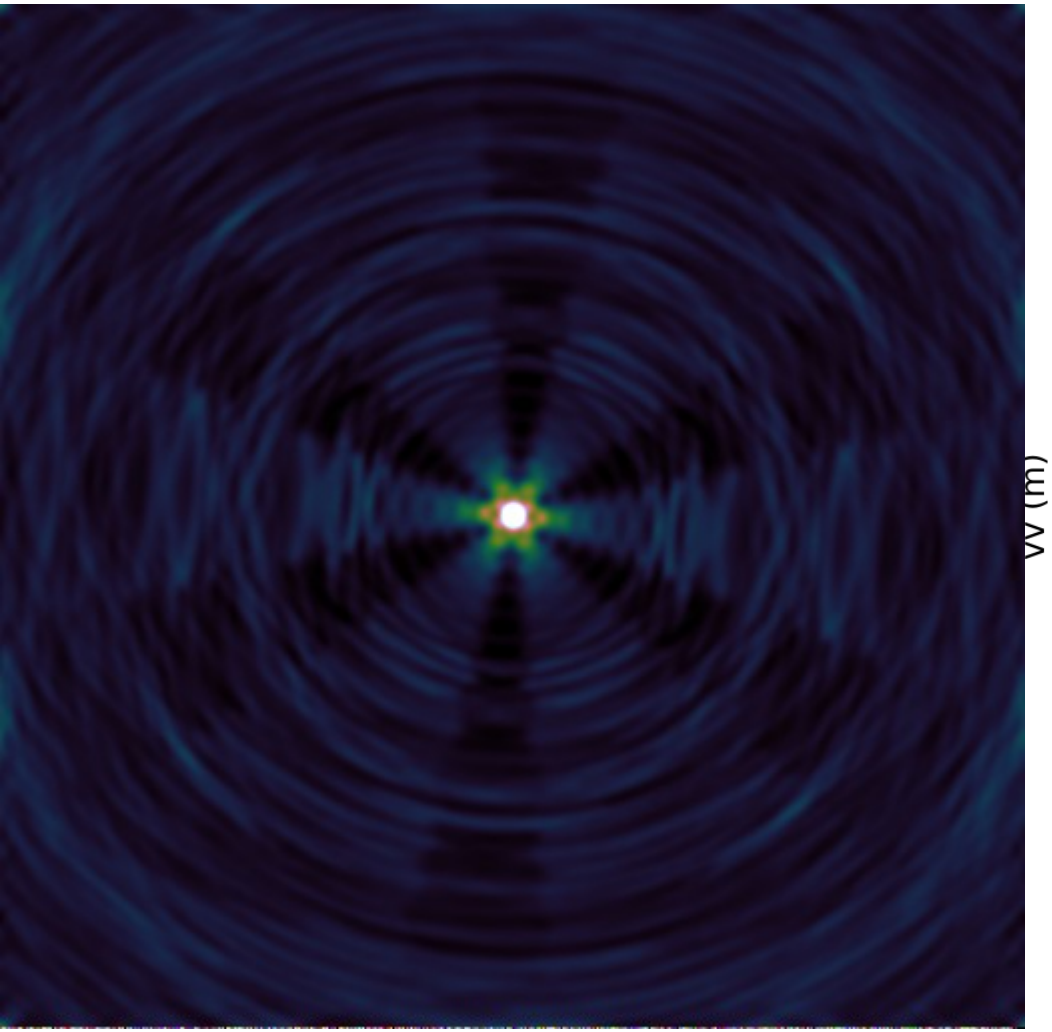


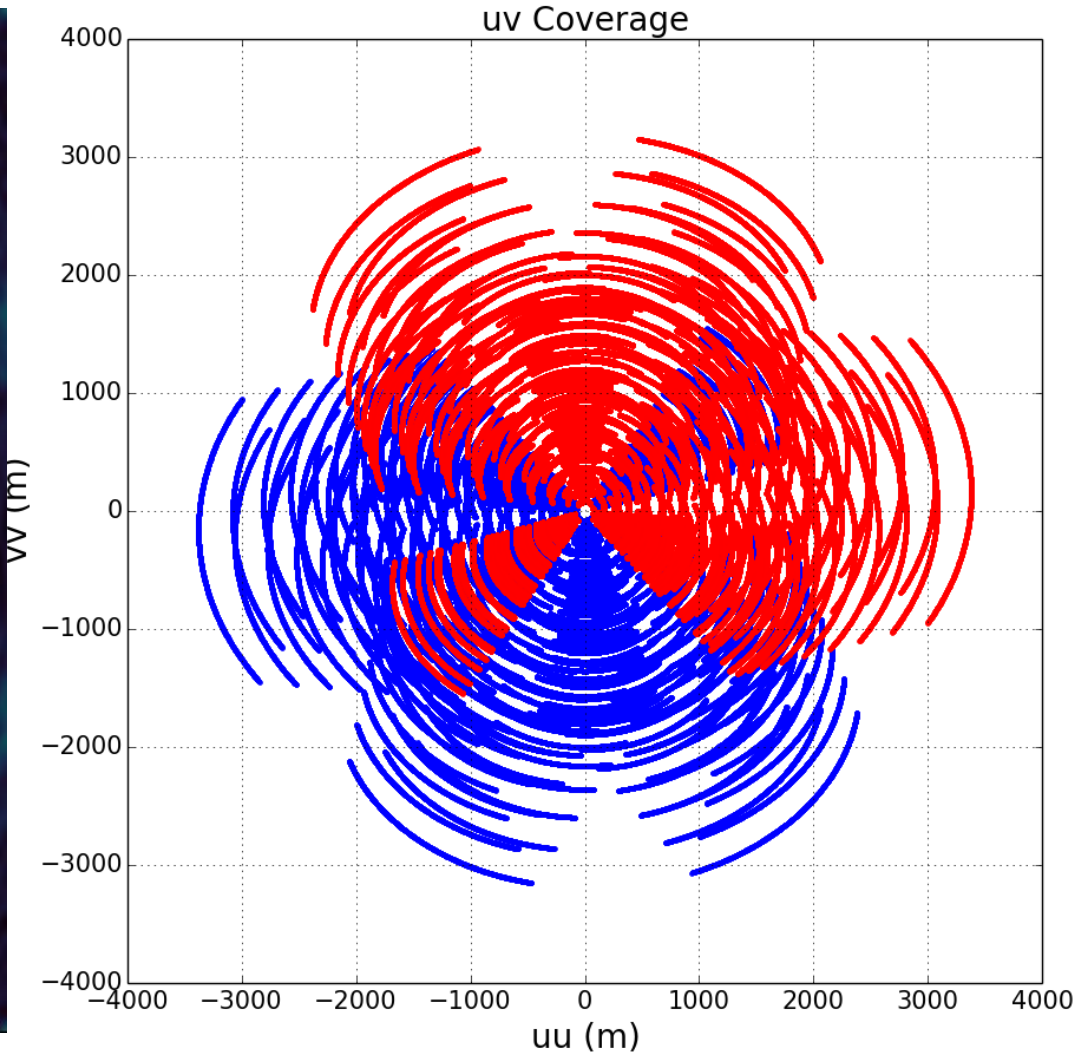
VLA PSF and uv Coverage
10 minutes (snapshot)



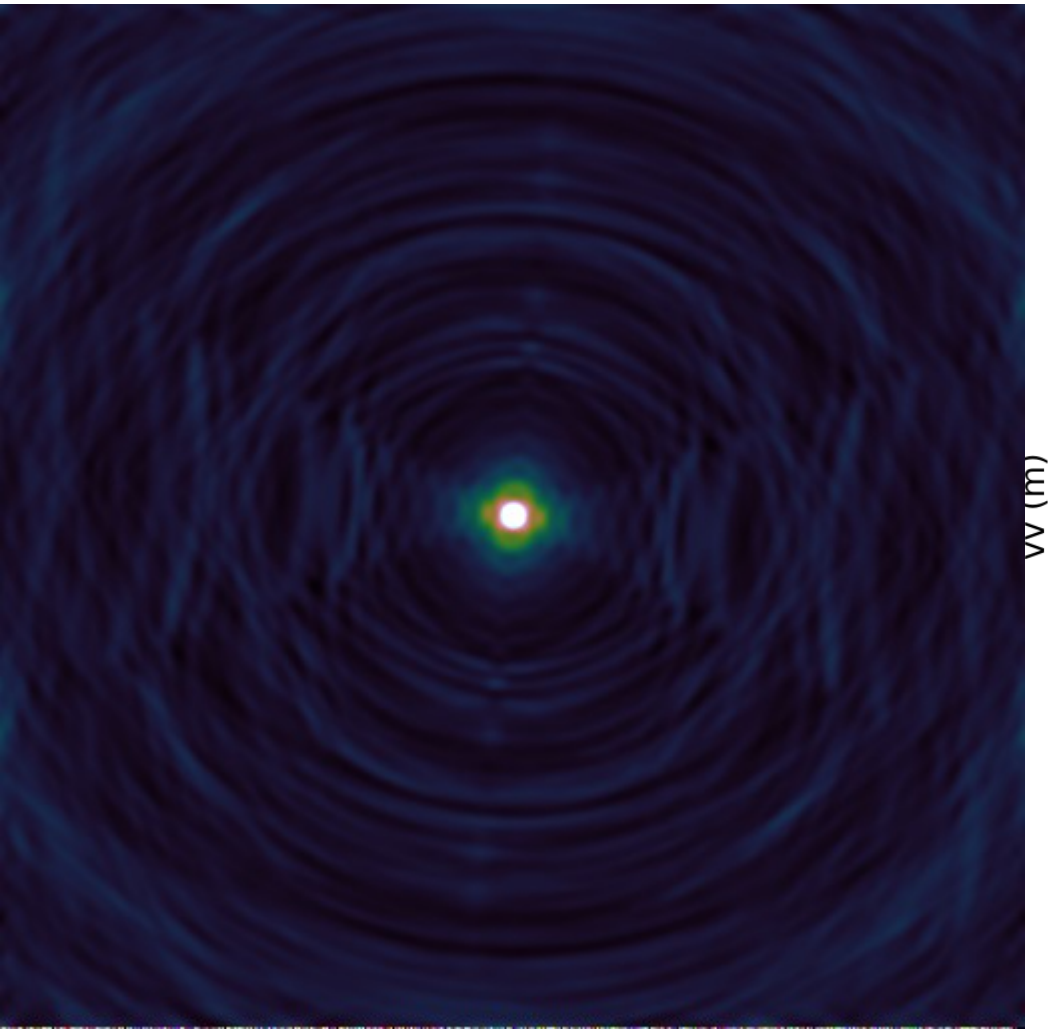


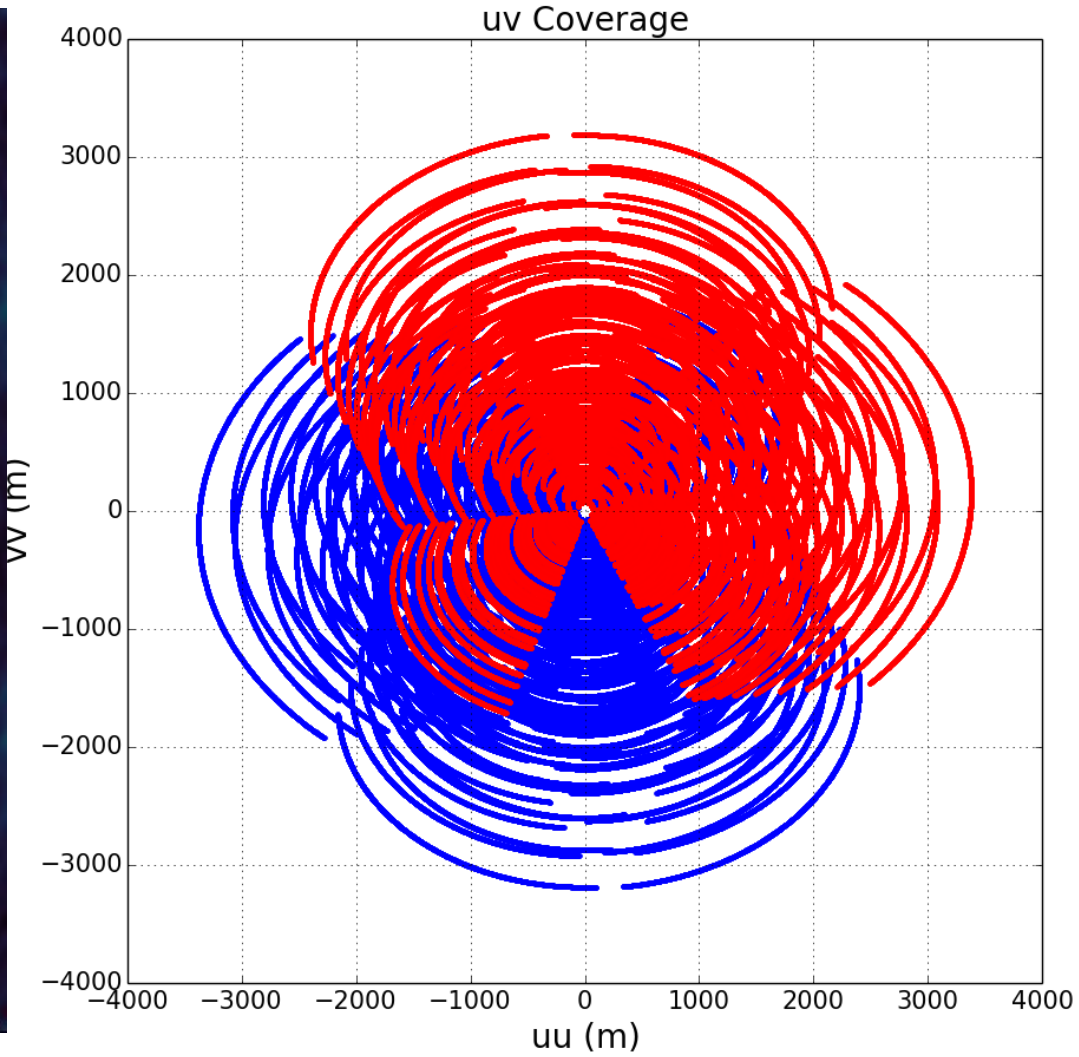
VLA PSF and uv Coverage
2 hours



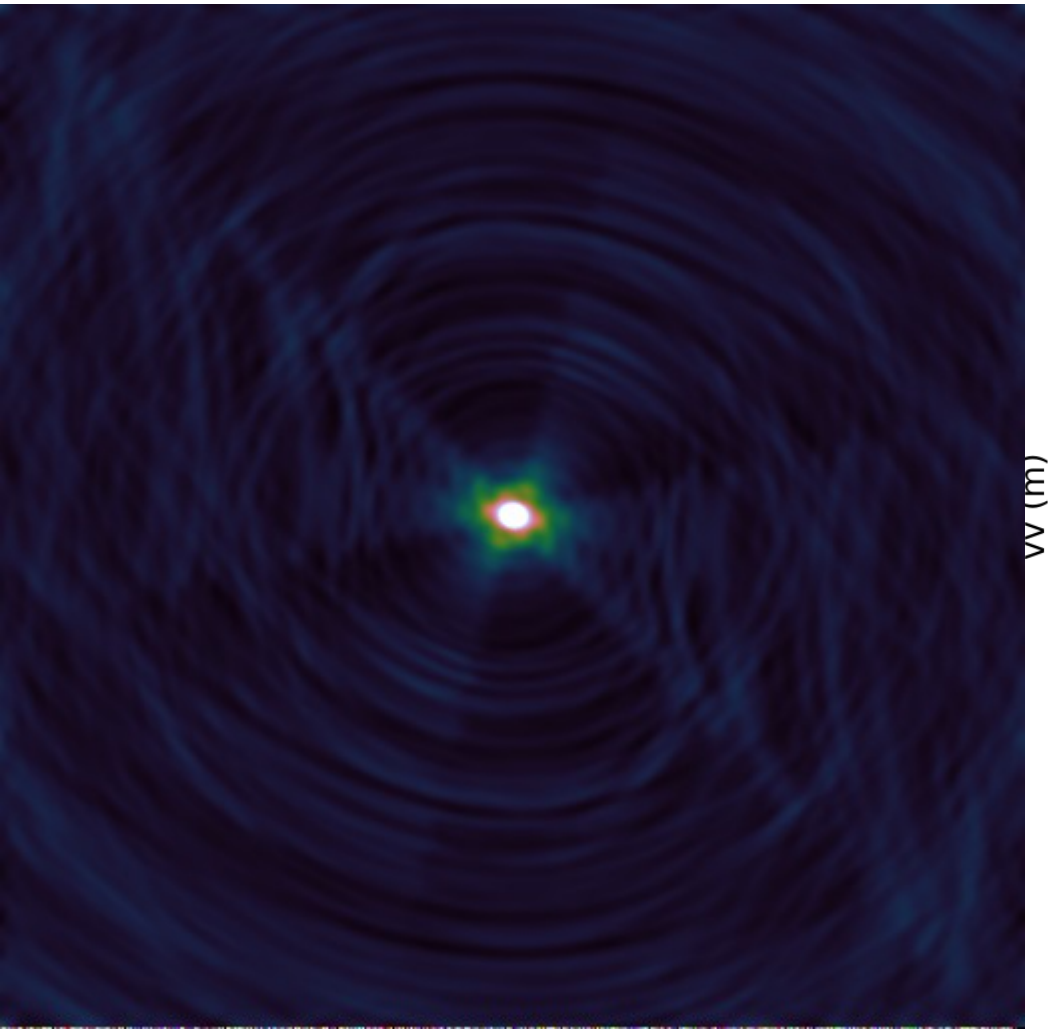


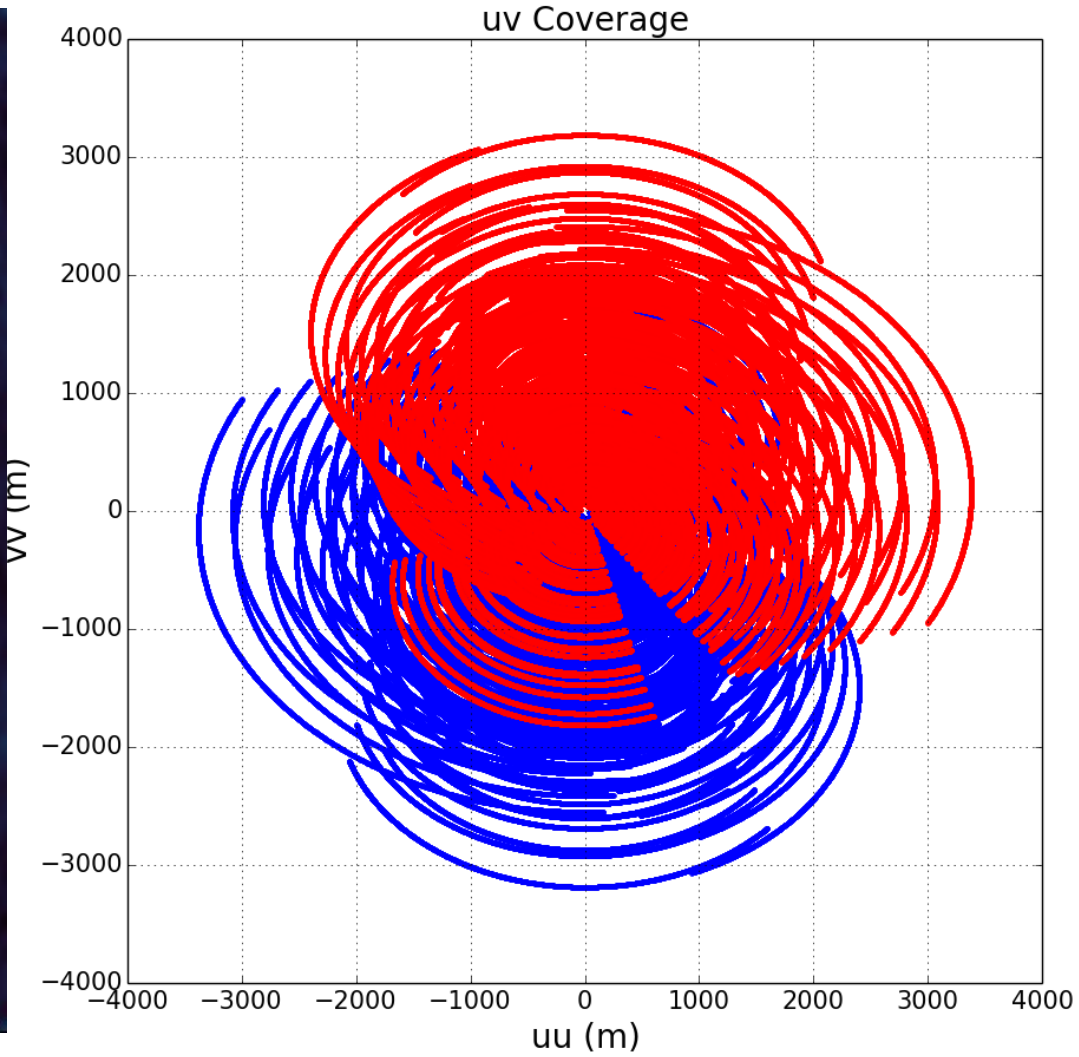
VLA PSF and uv Coverage
4 hours



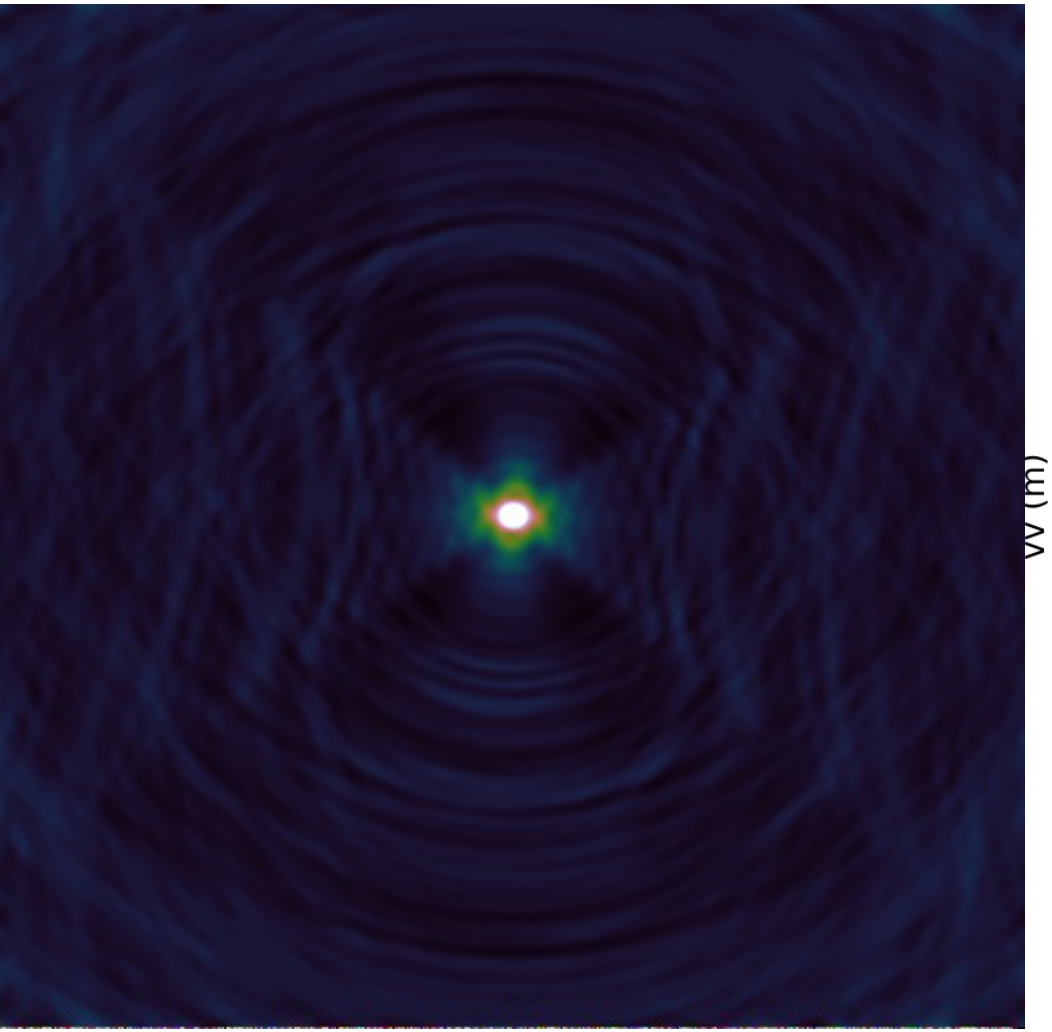


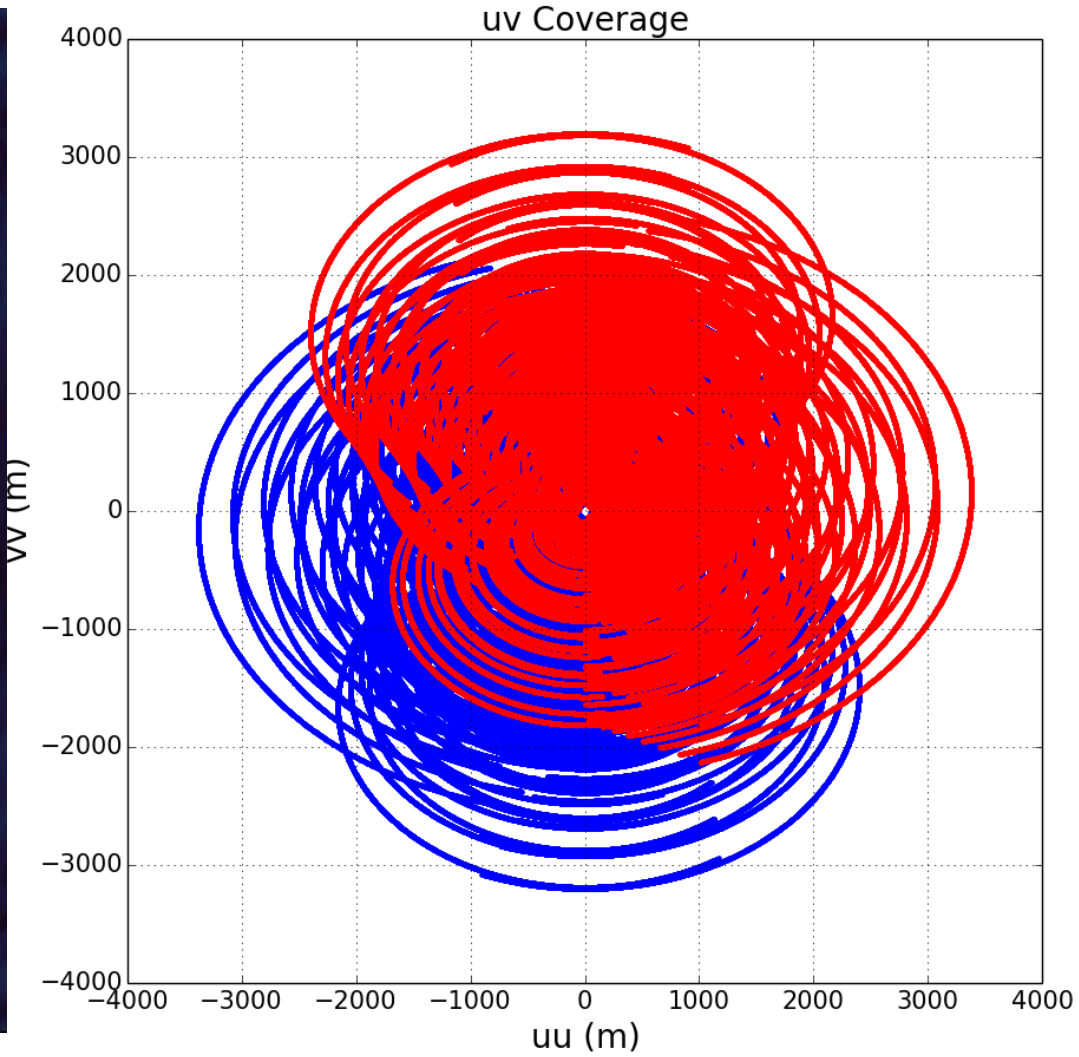
VLA PSF and uv Coverage
6 hours



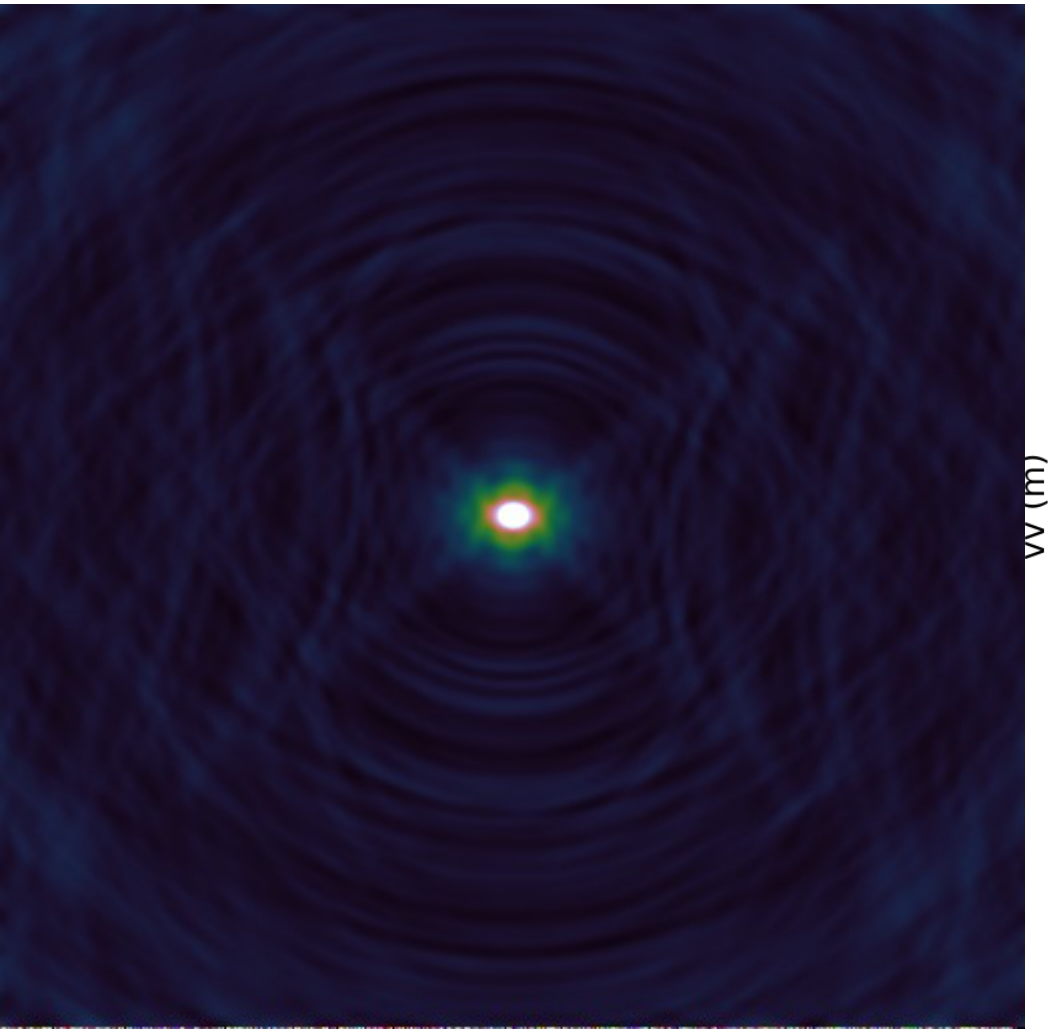


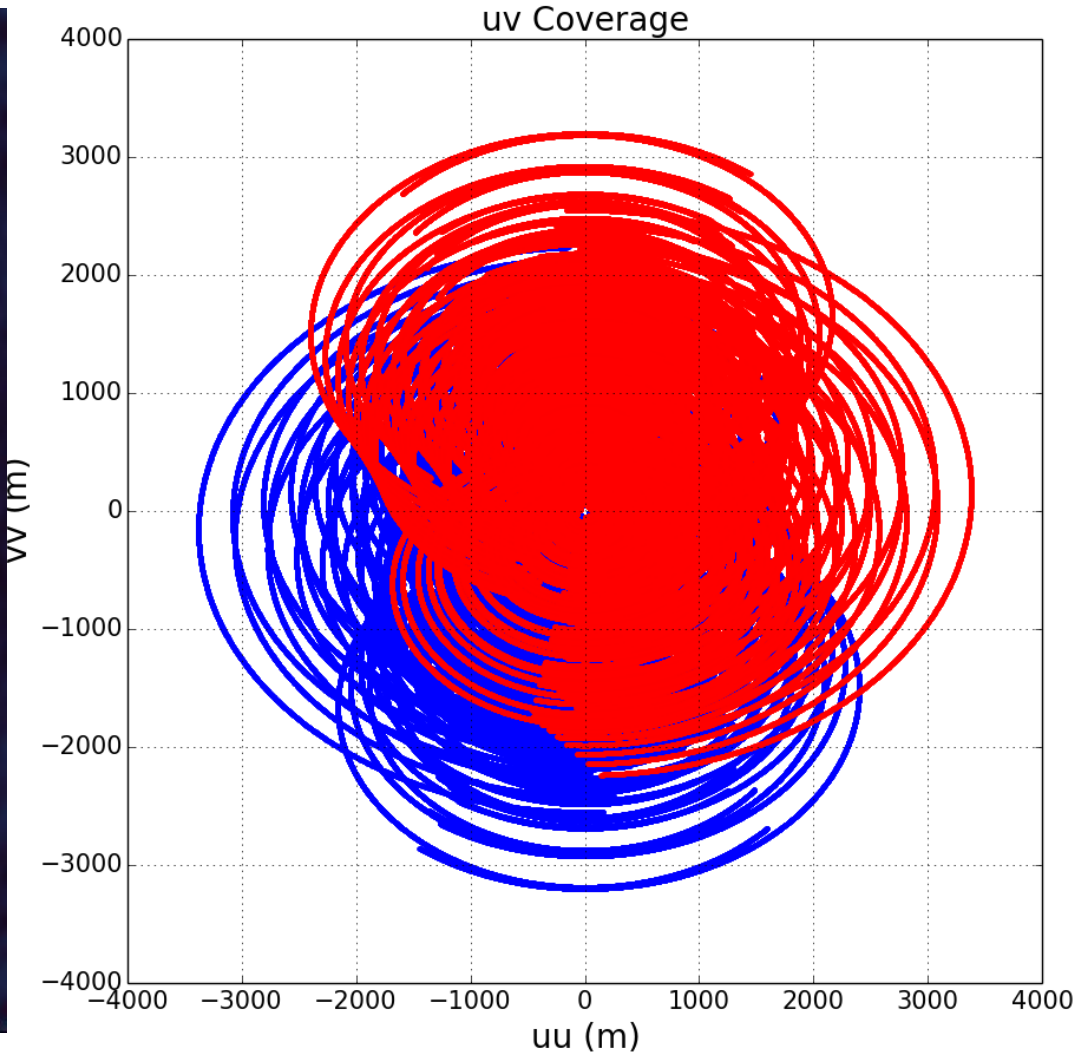
VLA PSF and uv Coverage
8 hours





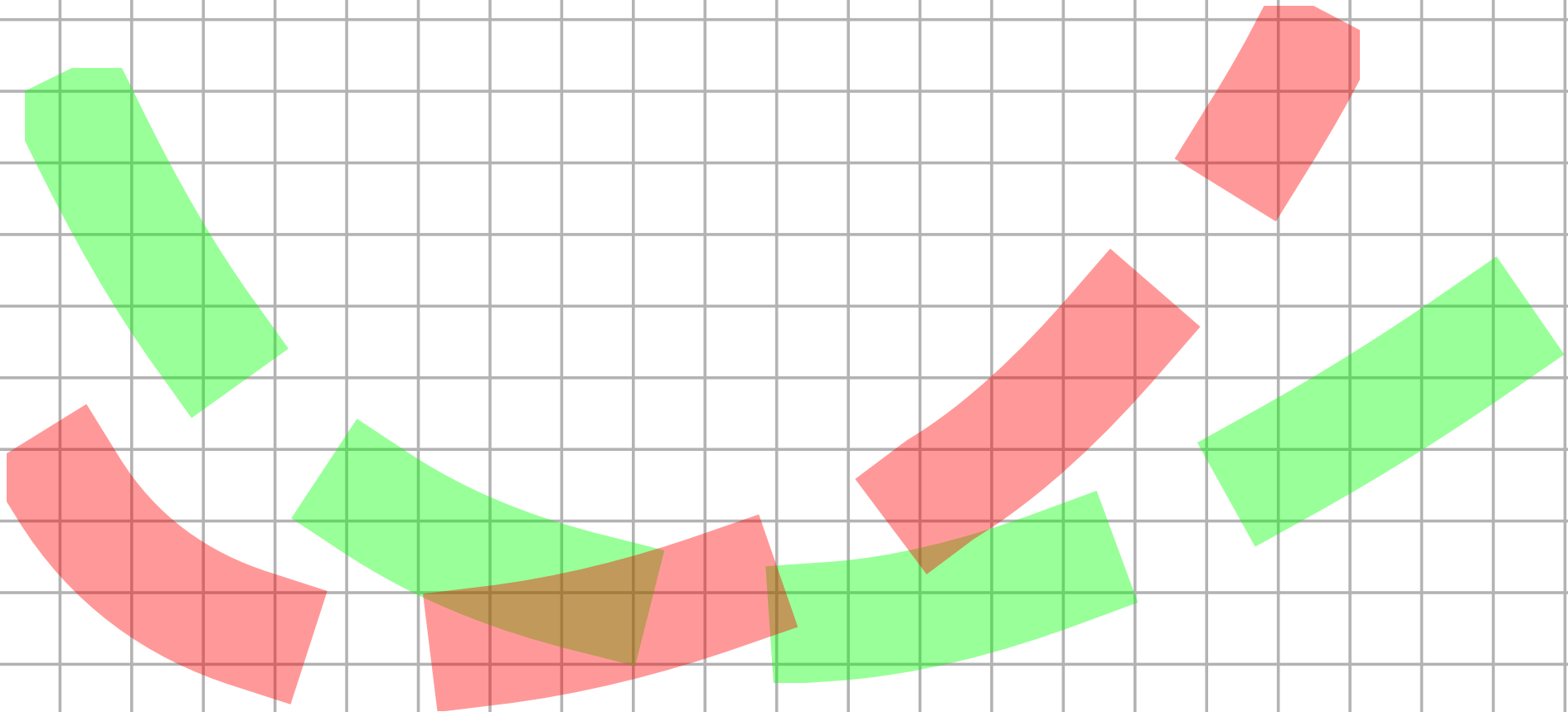
VLA PSF and uv Coverage
10 hours





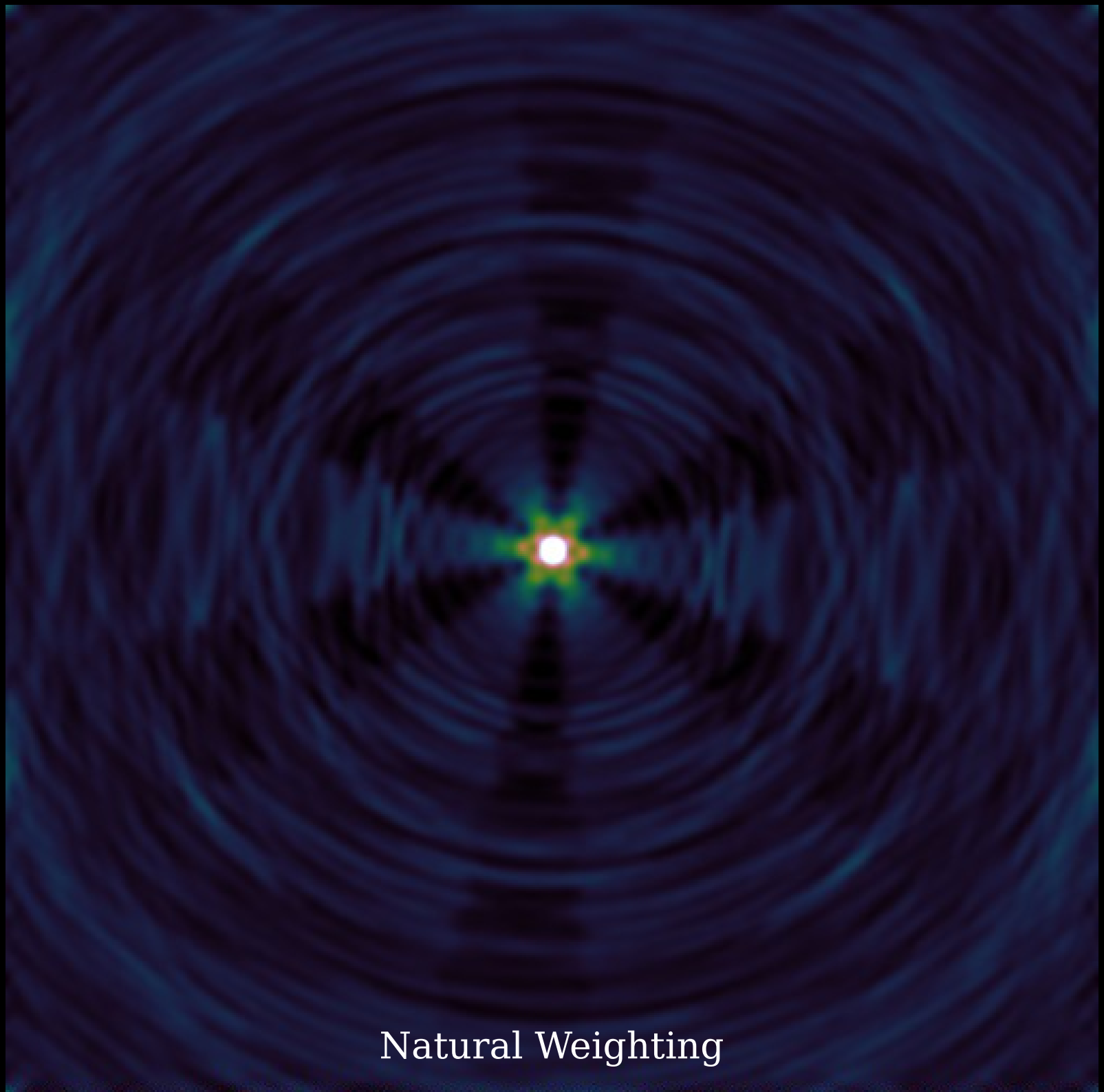
VLA PSF and uv Coverage
12 hours

When gridding visibilities to pixels what happens when there are overlapping visibility tracks?

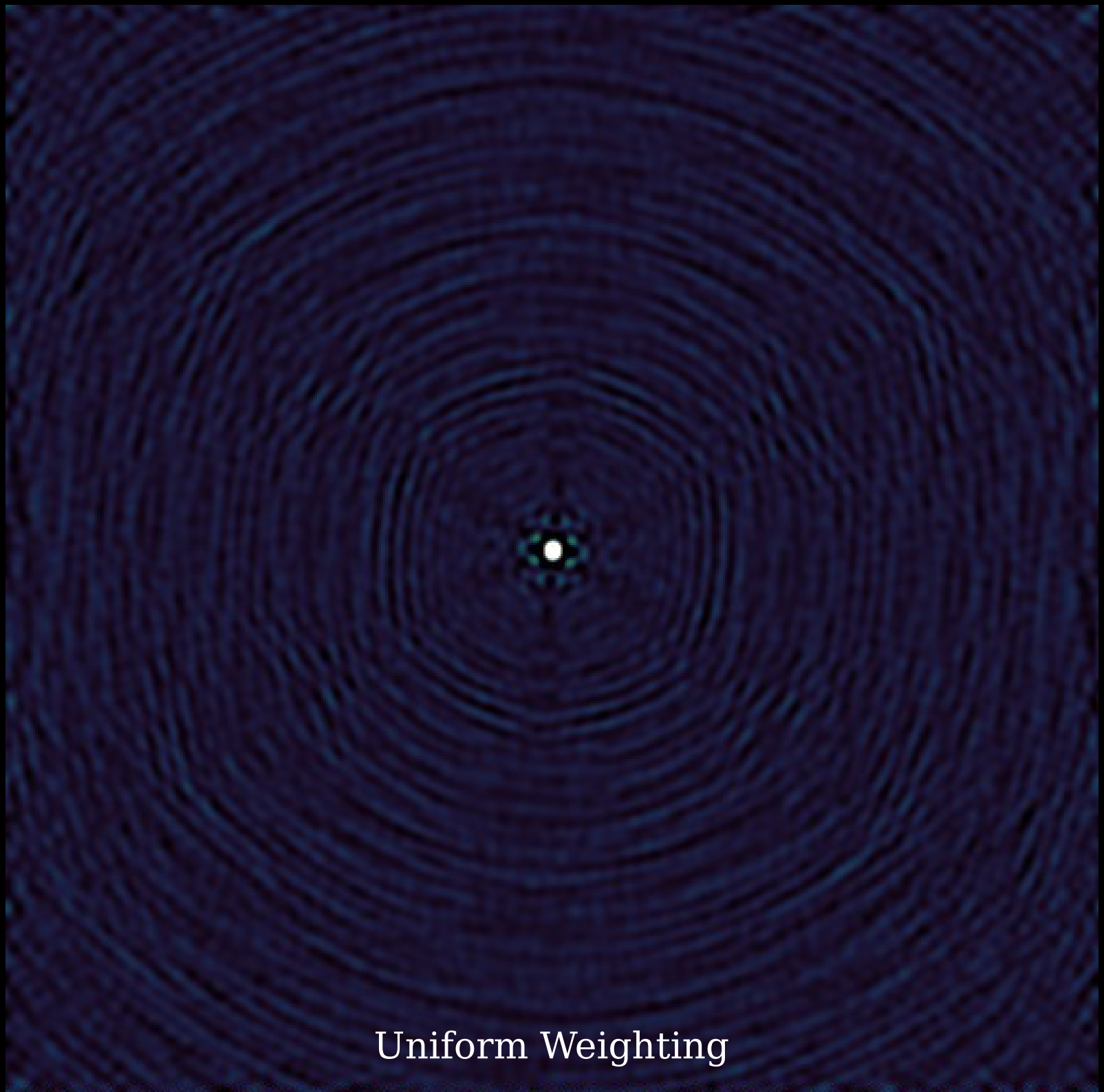


Weighting schemes trade off sensitivity for resolution:

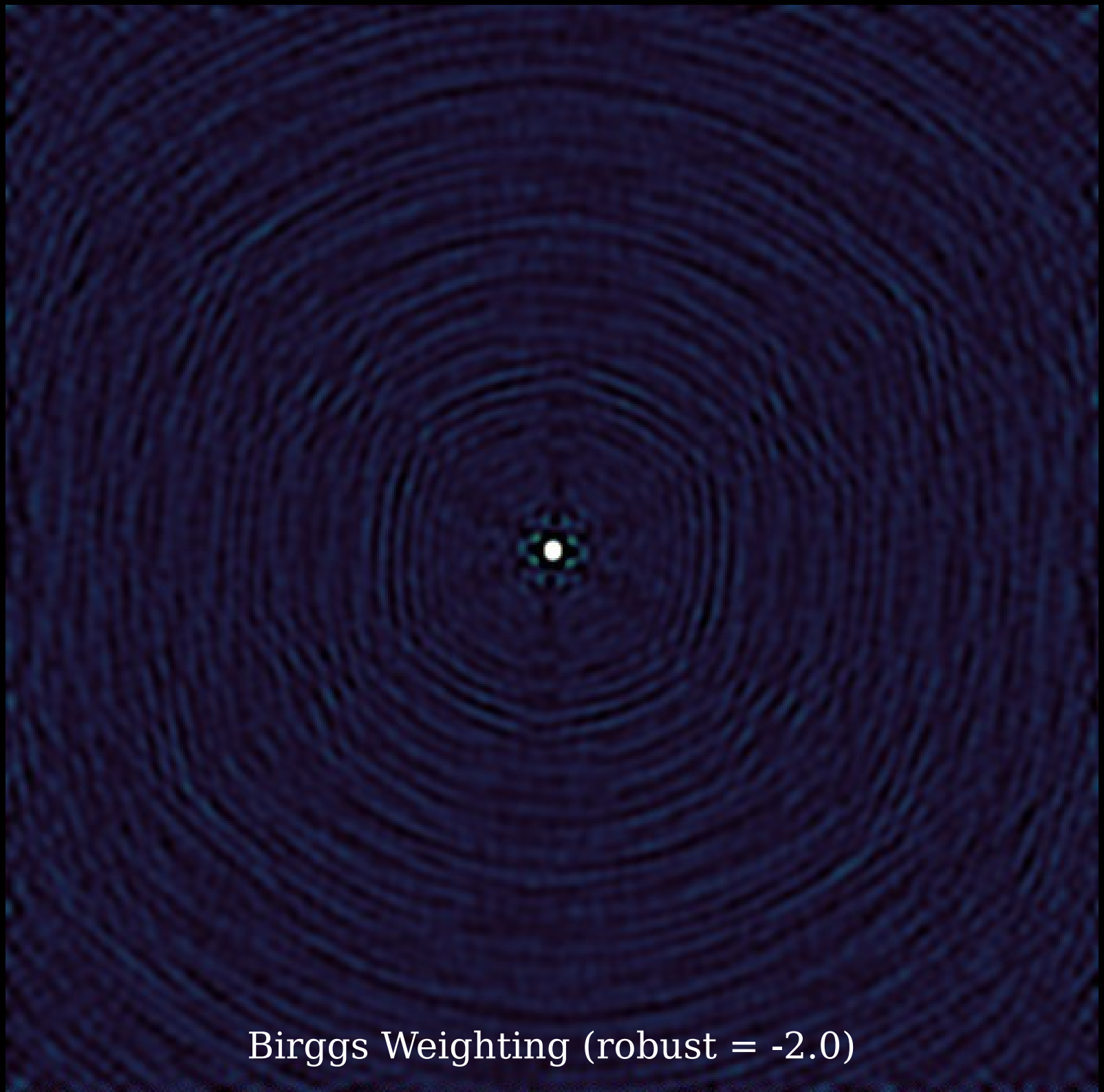
-Natural: each pixel is the sum of visibilities over that pixel (favour sensitivity)
-Uniform: each pixel is the average visibilities over that pixel (favour resolution)
-There are many other choices, e.g. Briggs/Robust, Super-uniform, Radial, ...



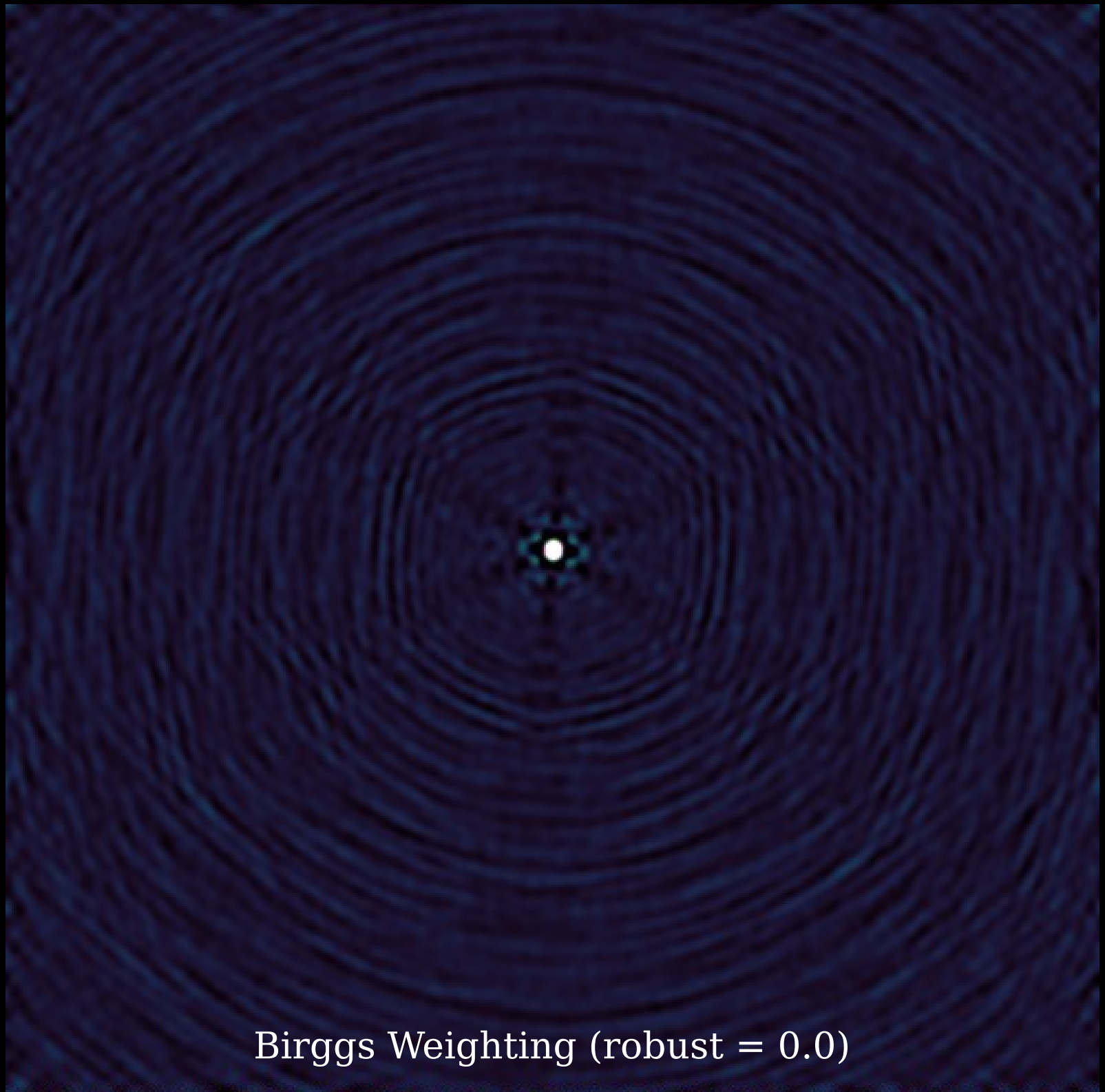
Natural Weighting



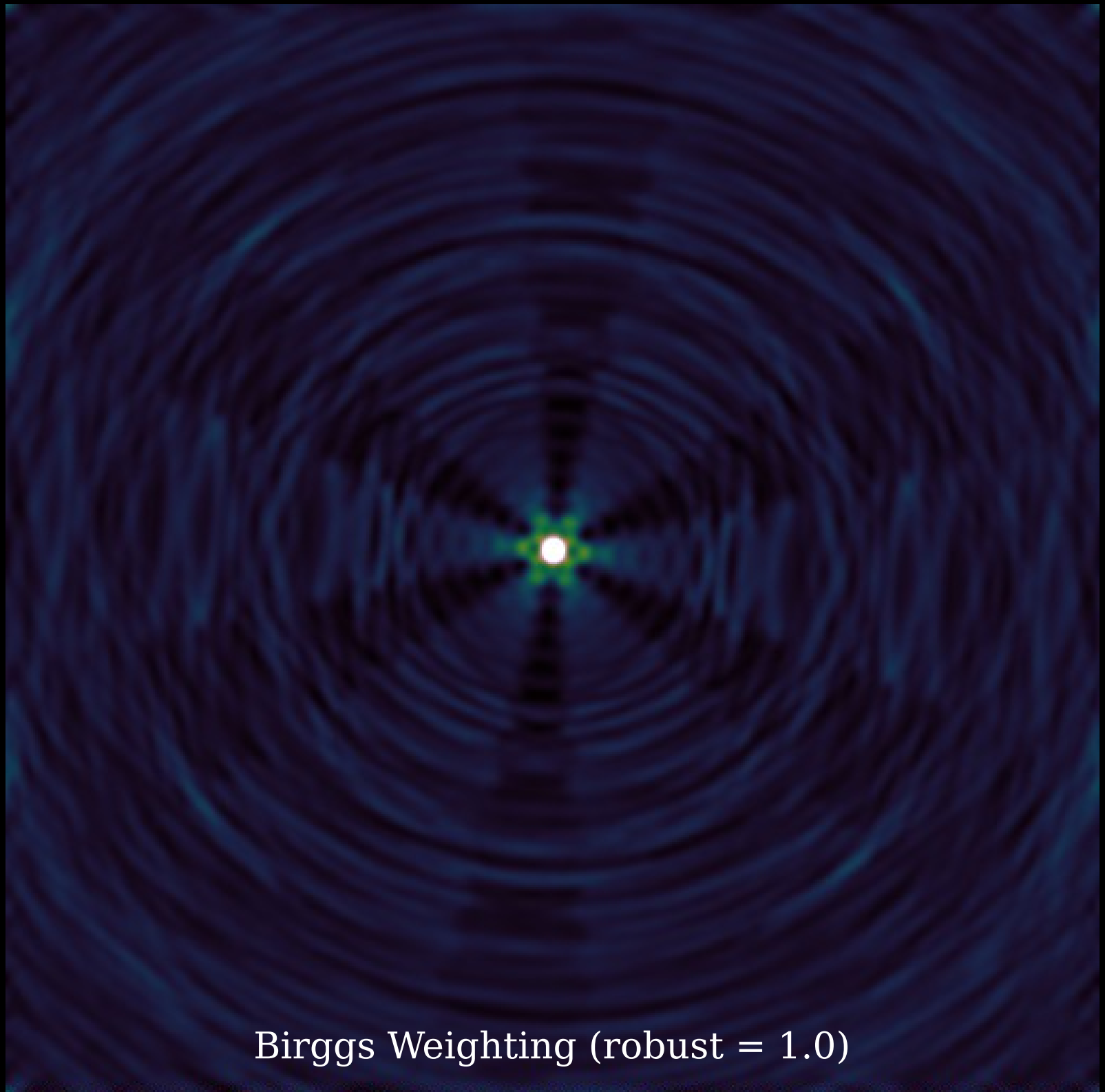
Uniform Weighting



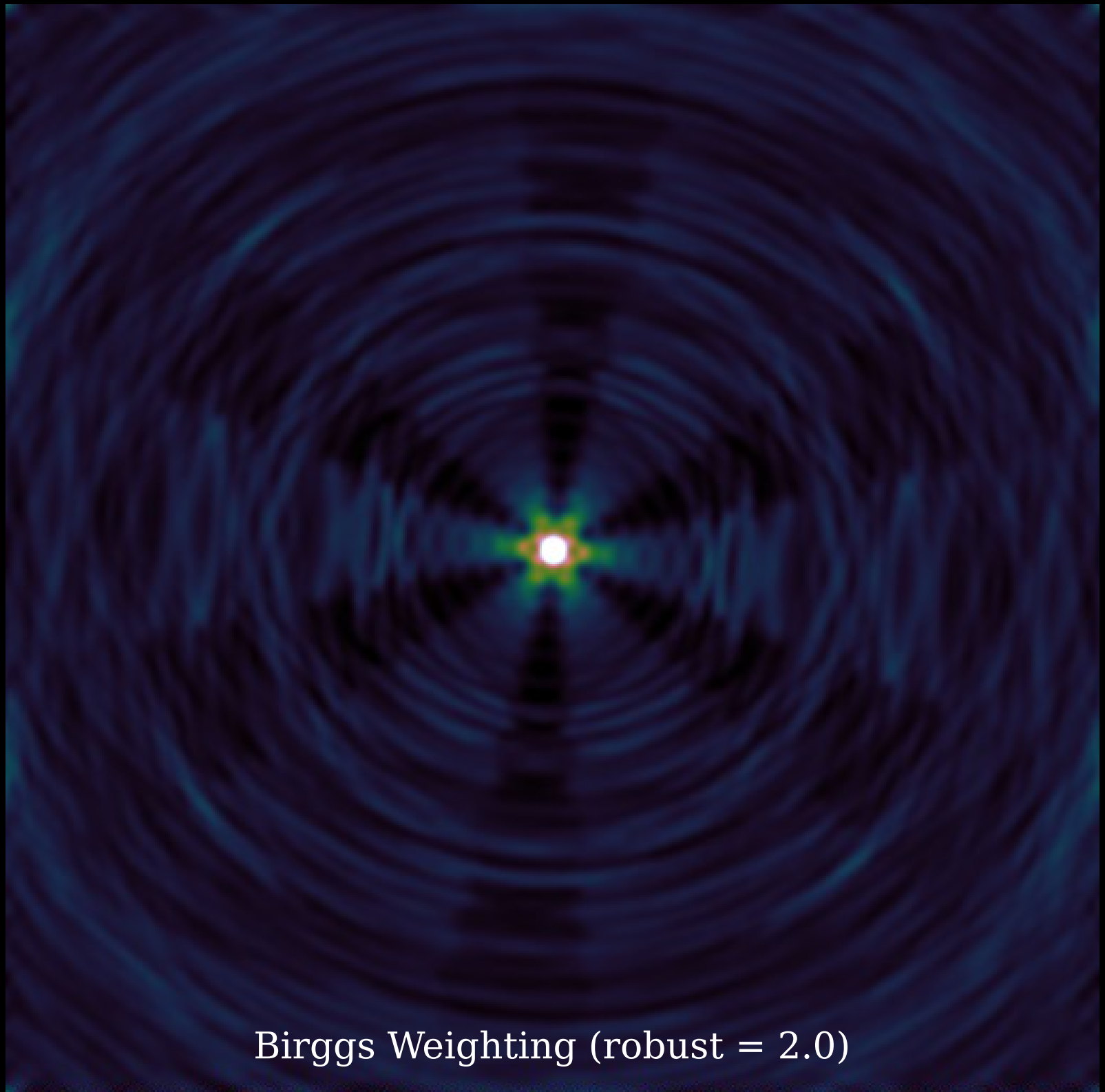
Birggs Weighting (robust = -2.0)



Birggs Weighting (robust = 0.0)

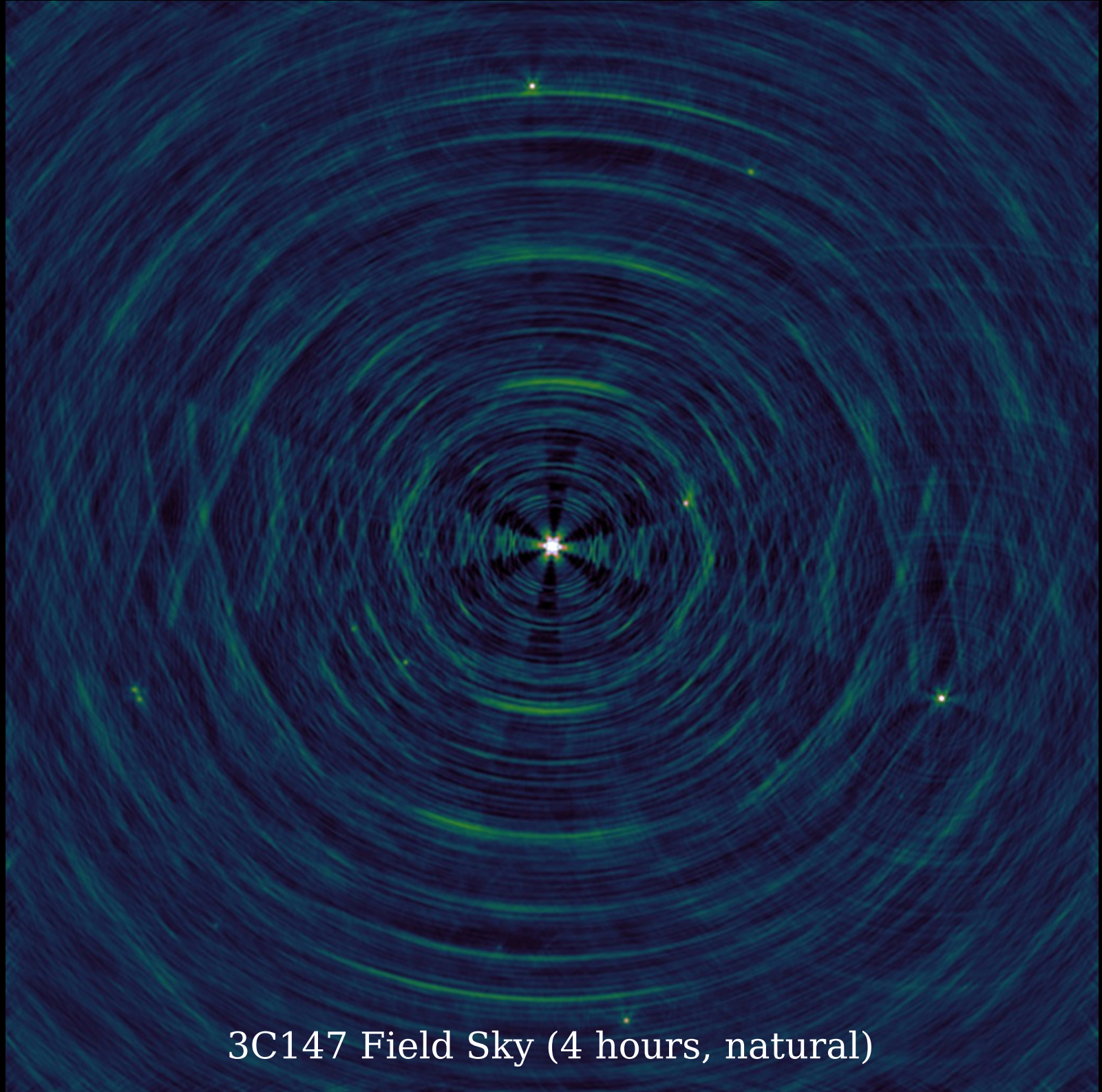


Birggs Weighting (robust = 1.0)

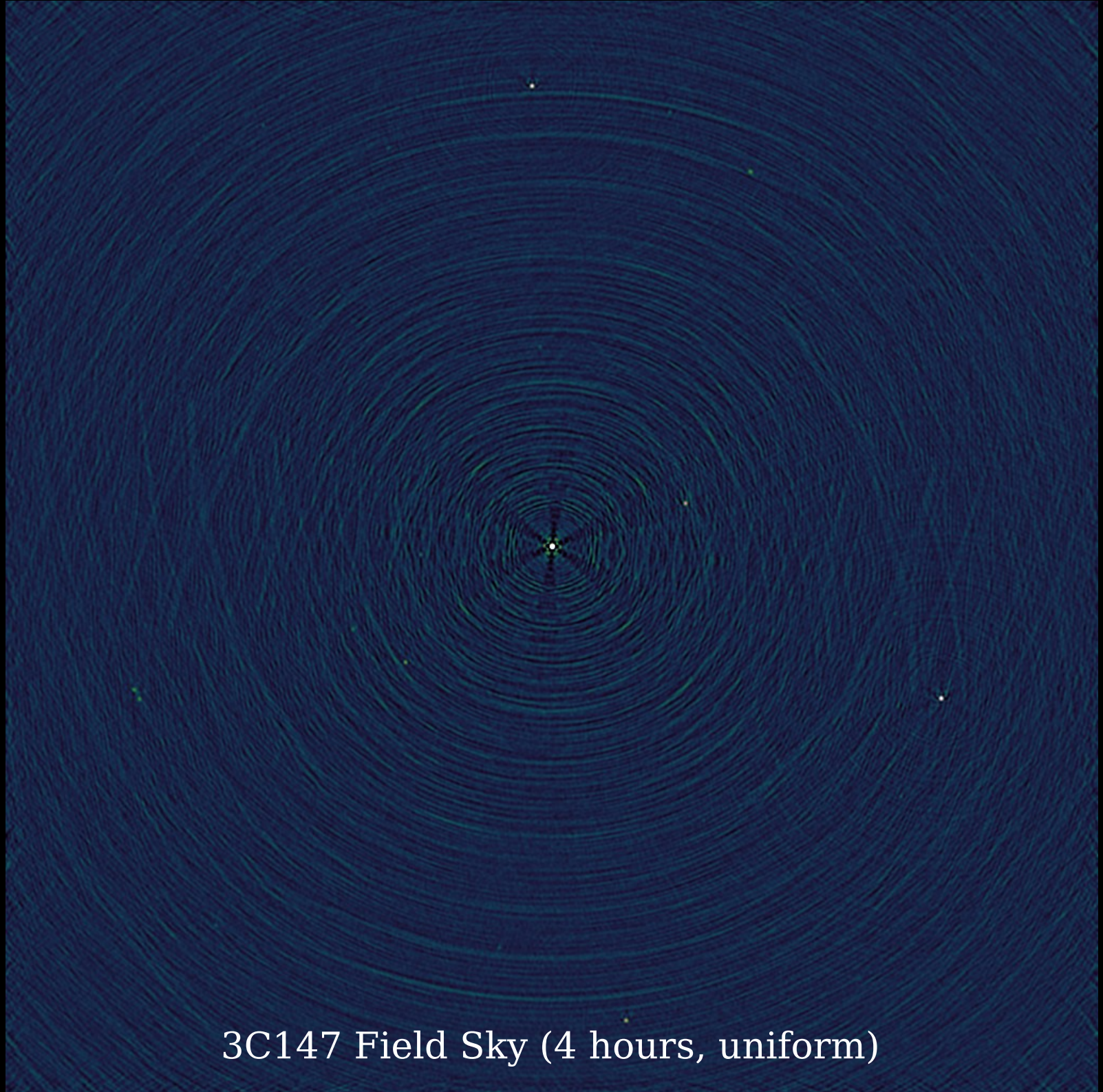


Birggs Weighting (robust = 2.0)

3C147 Field Sky (ideal)



3C147 Field Sky (4 hours, natural)



3C147 Field Sky (4 hours, uniform)

How to do we get to the sky model? 'Deconvolution'

Sky model: the ideal sky, usually represented as a collection of delta functions (point sources).

Point Spread Function (PSF): the response of the array to a point source.

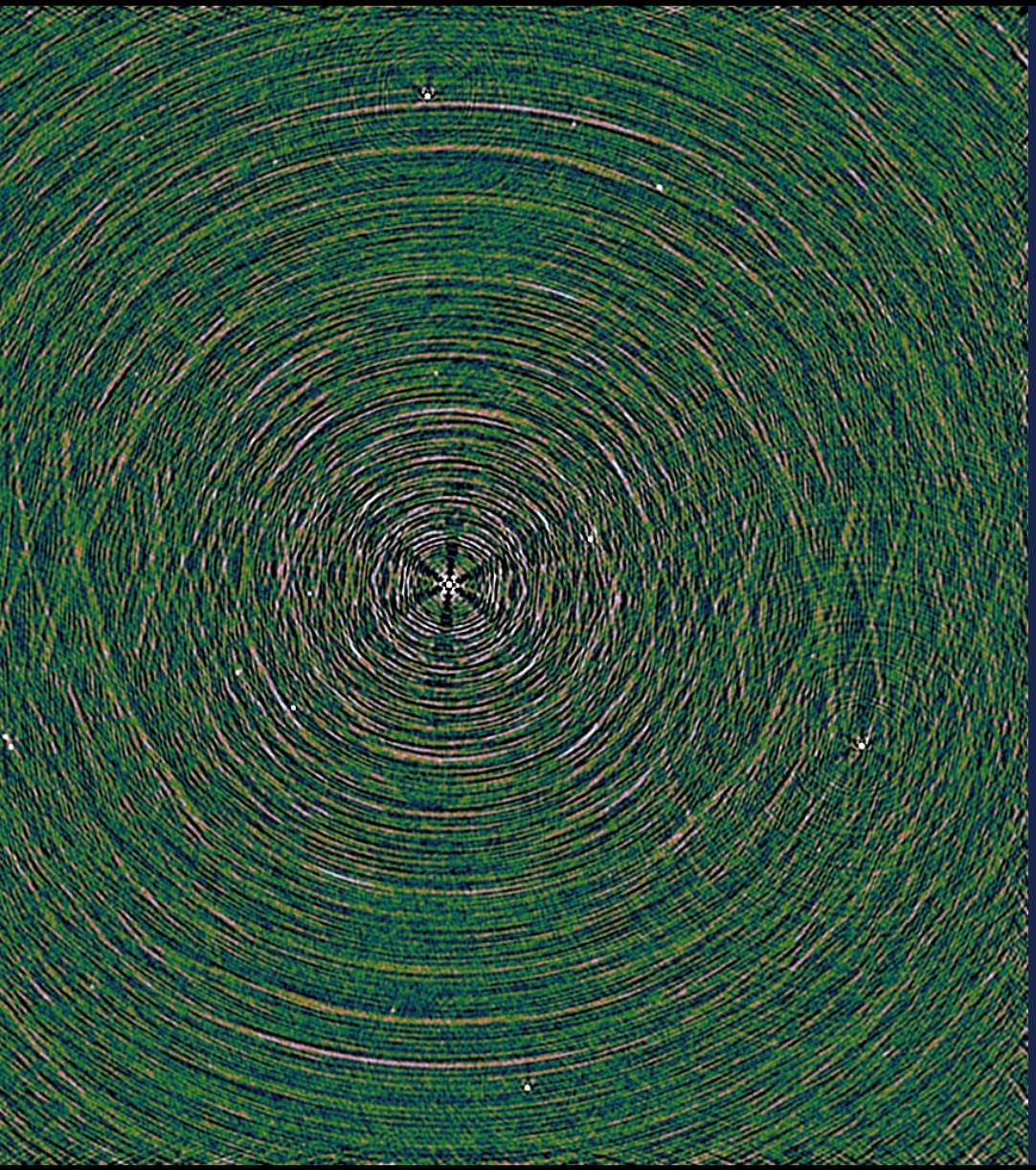
Dirty image: the image formed with the baselines visibilities using a weighting function. It is a convolution of the sky model, with the PSF response of the array.

Convolution is a simple process, but deconvolution is a bit more difficult. Usually, domain knowledge is required to aid in deconvolution. In the case of interferometry we often use the assumption that everything in the sky model is a point source.

Residual image: the remaining flux distribution after a deconvolution is performed.

Clean/restored image: the residual image with the sky model convolved with an ideal PSF.

3C147 Field Sky (4 hours, uniform)

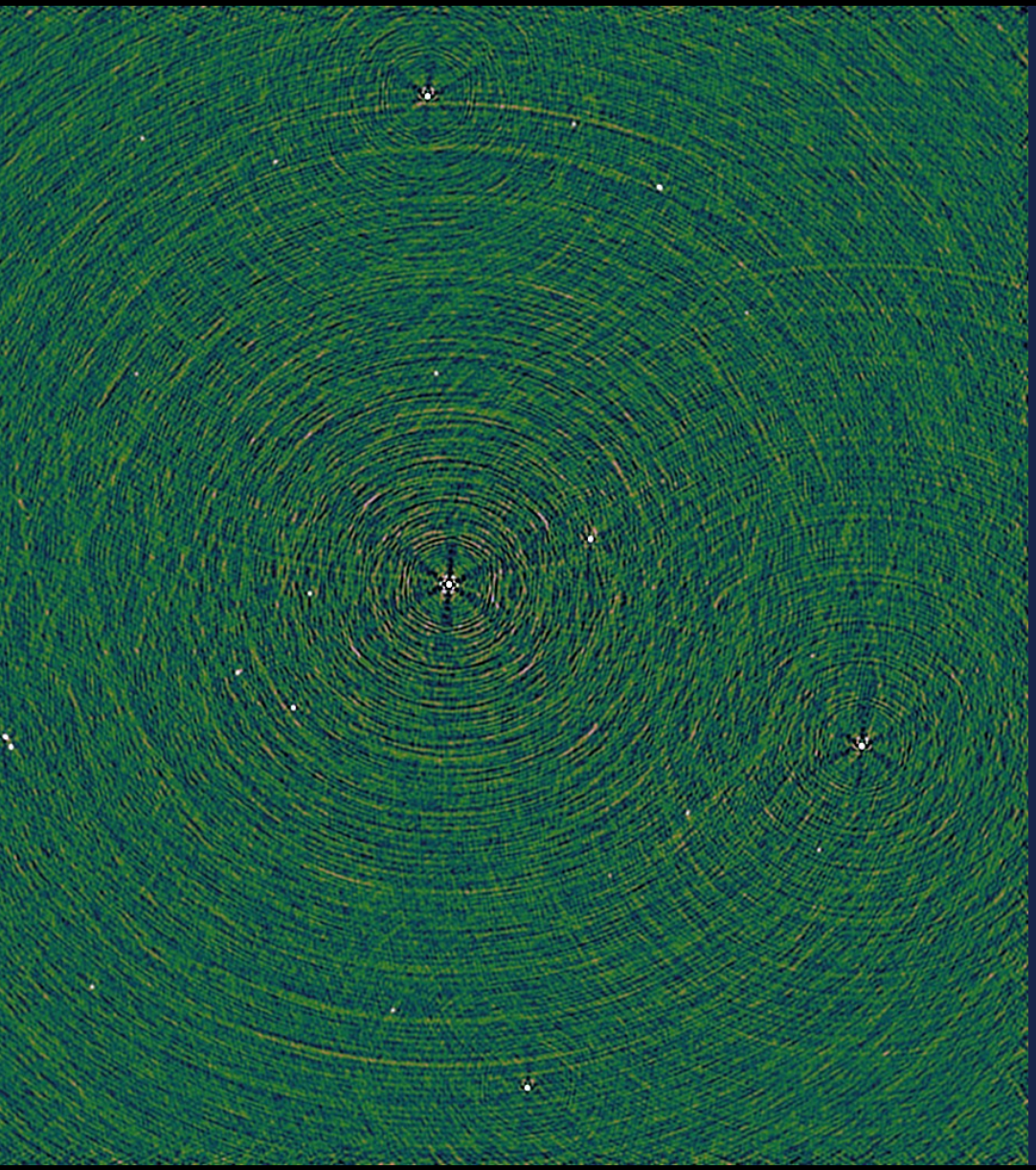


Residual Image

Derived Model Sky

CSCLEAN : 1 iteration, 0.1 gain

3C147 Field Sky (4 hours, uniform)

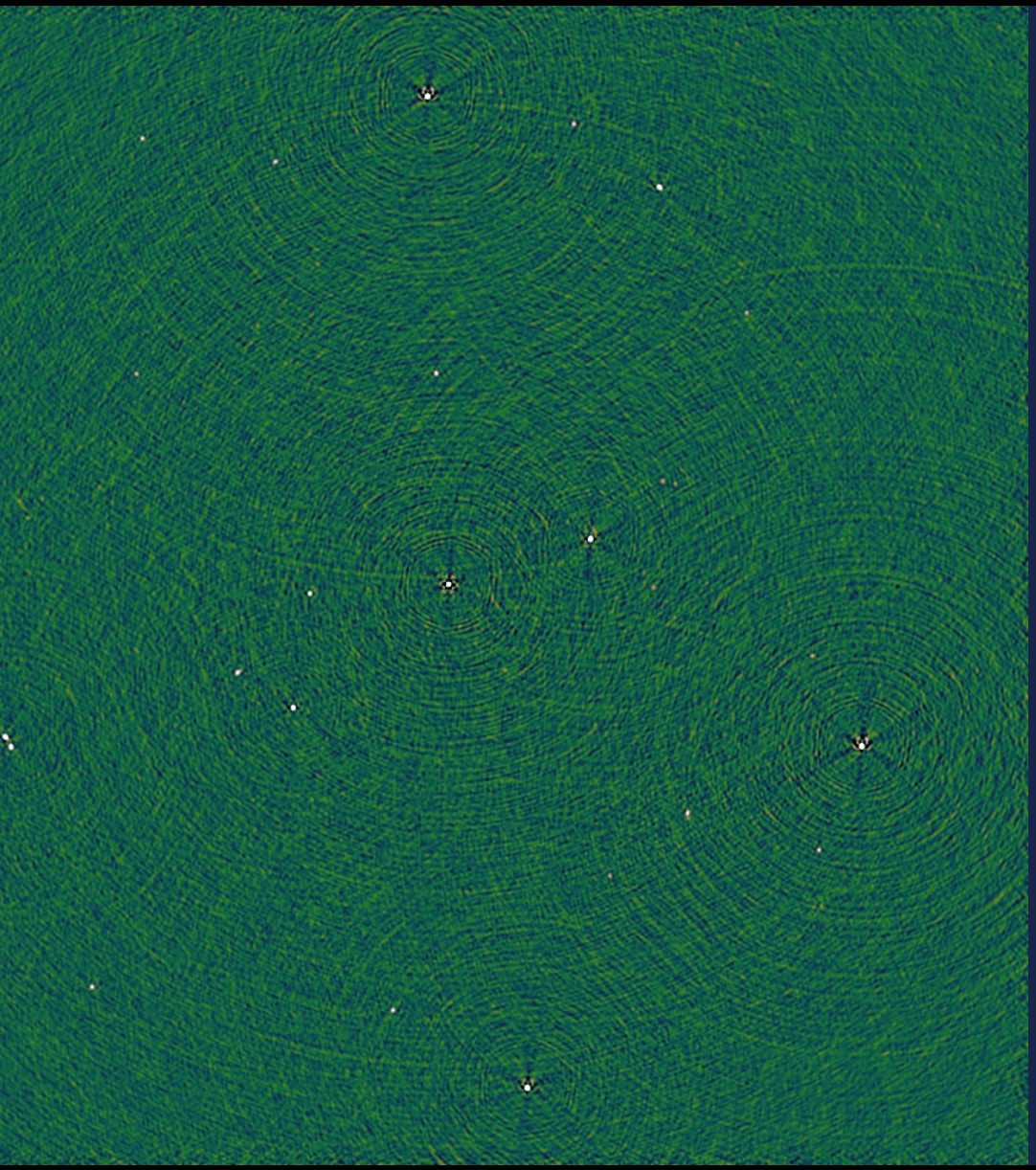


Residual Image

Derived Model Sky

CSCLEAN : 10 iteration, 0.1 gain

3C147 Field Sky (4 hours, uniform)

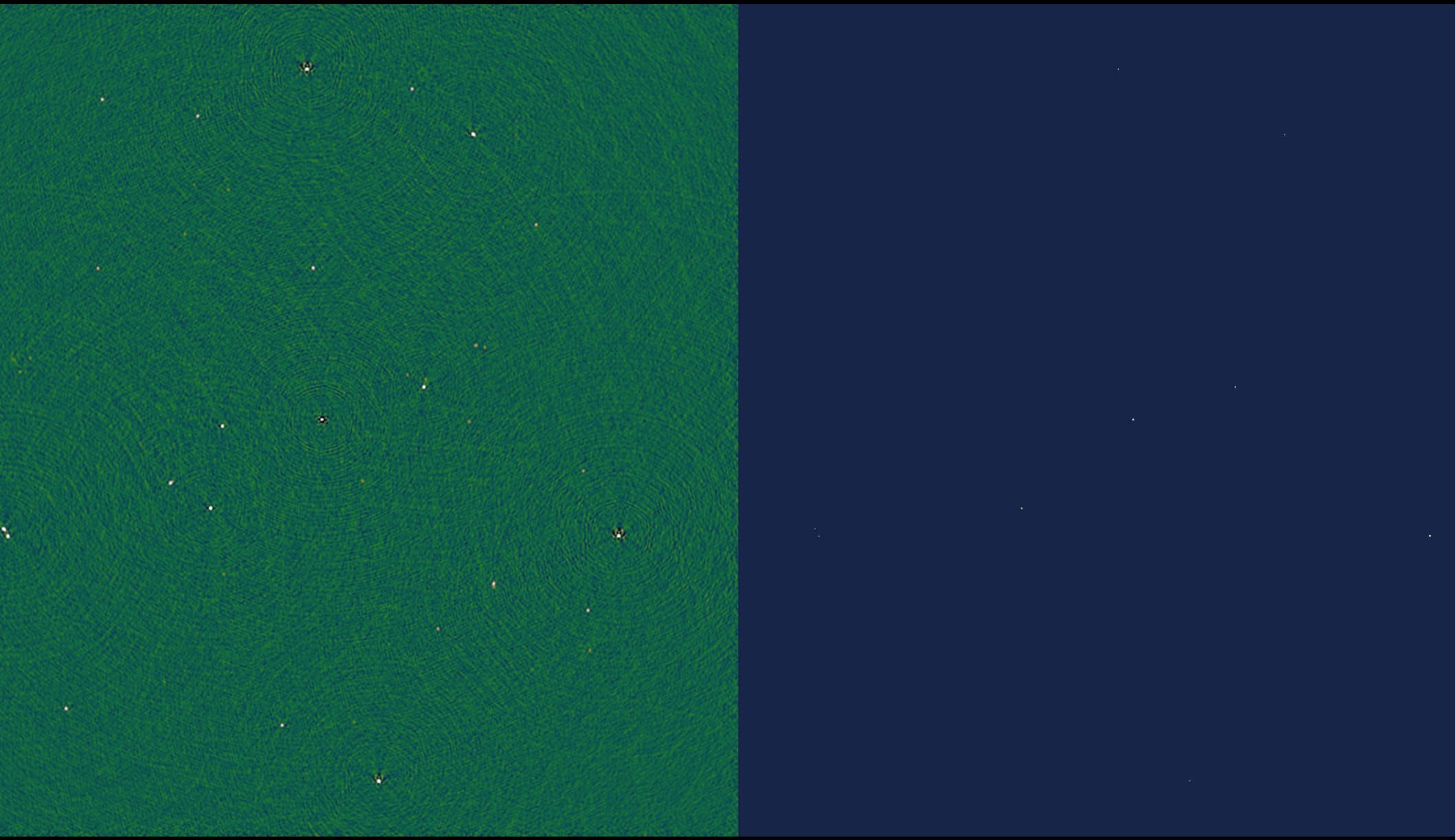


Residual Image

Derived Model Sky

CSCLEAN : 30 iteration, 0.1 gain

3C147 Field Sky (4 hours, uniform)

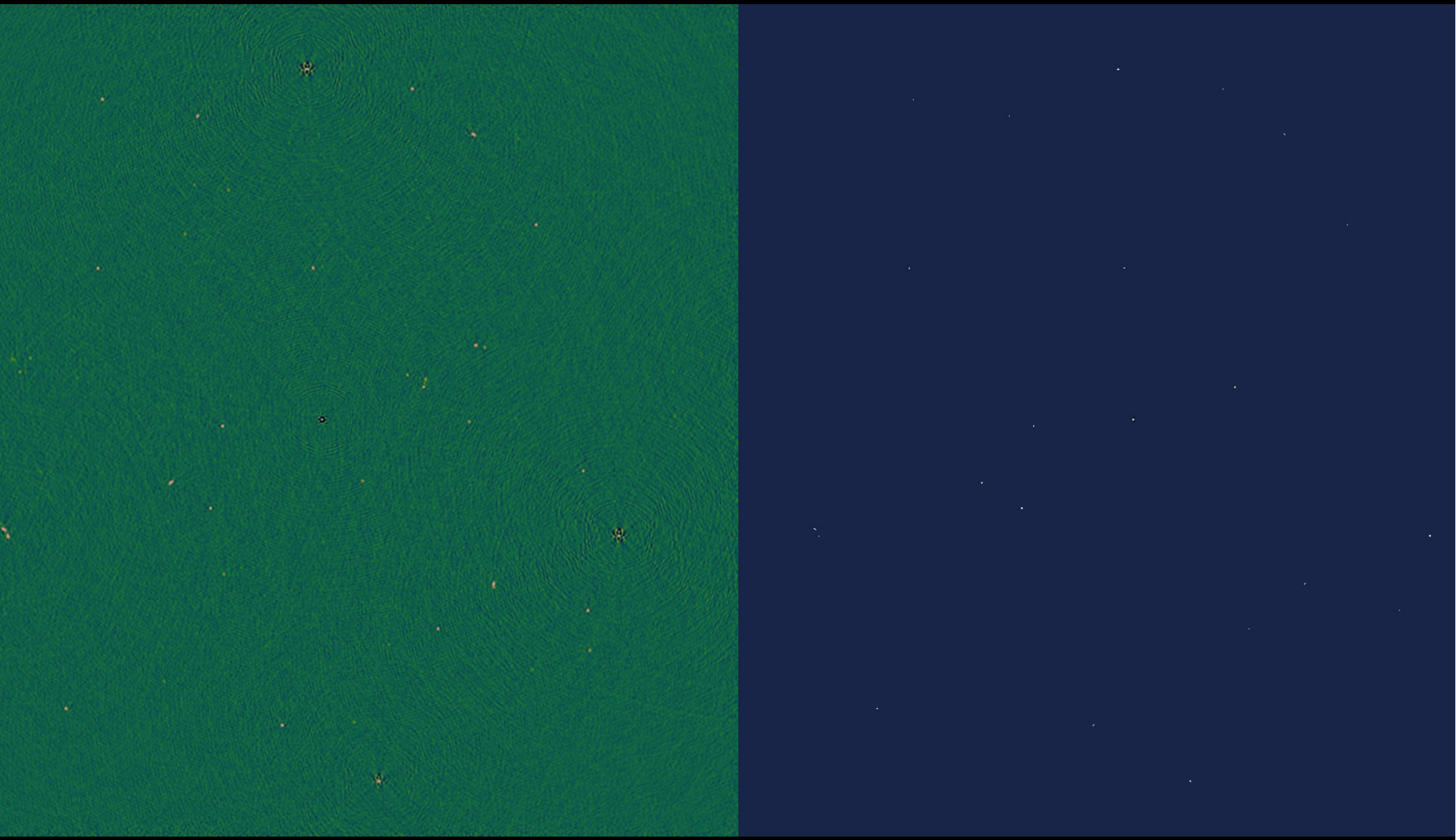


Residual Image

Derived Model Sky

CSCLEAN : 100 iteration, 0.1 gain

3C147 Field Sky (4 hours, uniform)

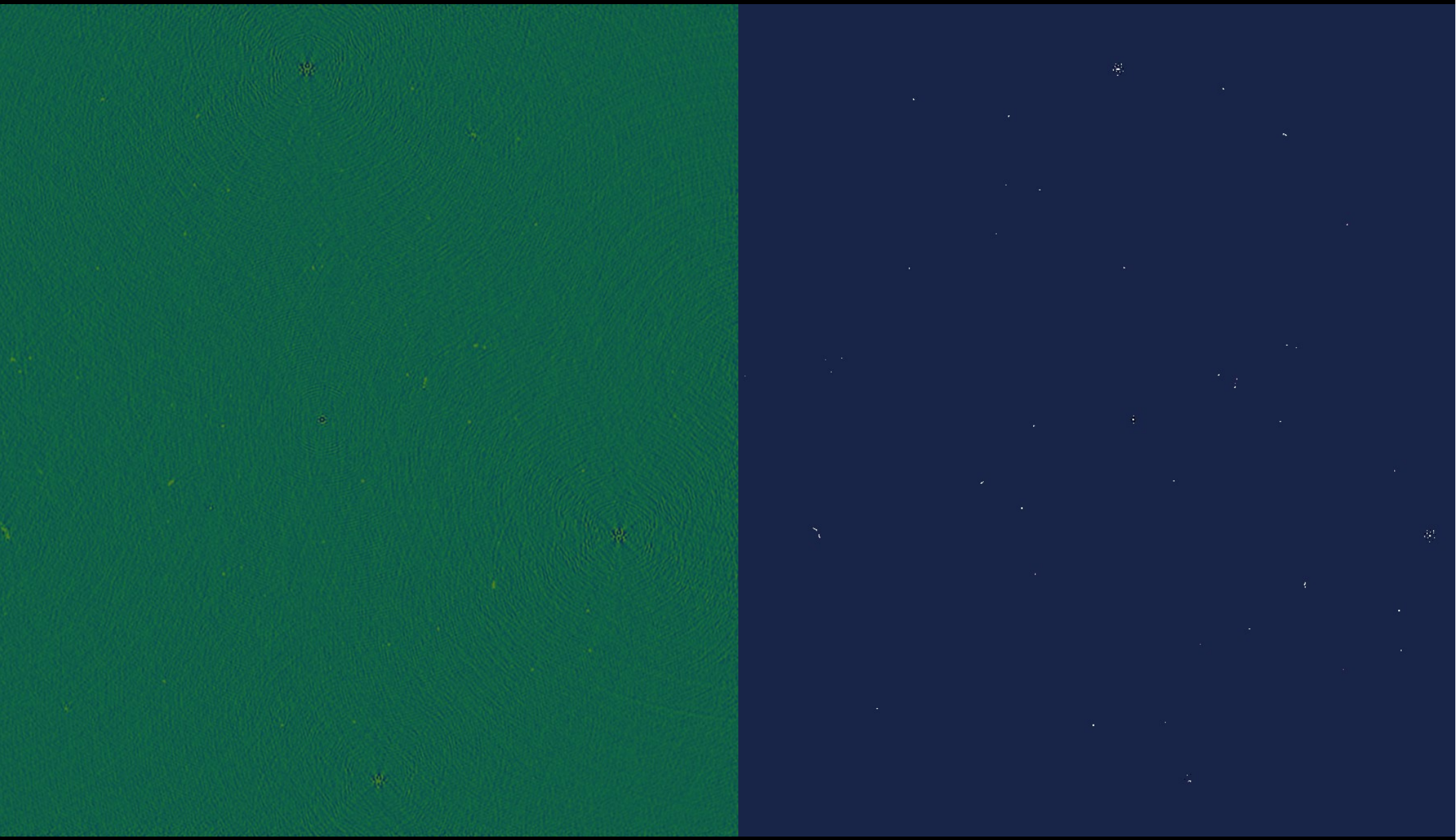


Residual Image

Derived Model Sky

CSCLEAN : 300 iteration, 0.1 gain

3C147 Field Sky (4 hours, uniform)



Residual Image

Derived Model Sky

CSCLEAN : 1000 iteration, 0.1 gain

Högbom's Algorithm (Image-domain CLEAN):

Inputs: Dirty Image, PSF
Parameters: gain (γ), end threshold

1) Make a copy of Dirty Image, call it the Residual Image
Create an empty Model

2) while(End threshold not reached):
    a) Find the pixel p_{max} with the largest flux f_{max} in the Residual Image
    b) Subtract γ*PSF centred on p_{max} and update the Residual Image
    c) In the Model, save the position p_{max} and amplitude $\gamma * f_{max}$ of the subtracted component

3) Convolve the Model with the 'restored' beam, and add it to the final Residual Image to produce the Restored Image

Output: Model, Residual Image, Restored Image

Gain: less than 1, usually 0.1, the portion of flux in a pixel that is removed in each iteration

End threshold: either a maximum number of iterations or a remaining flux per pixel is below a minimum level

Clark's Algorithm (Gridded-visibilities CLEAN):

Inputs: Dirty Image, PSF
Parameters: gain (γ), end threshold

A) Make a copy of Dirty Image, call it the Residual Image
B) Create an empty Model

C) while(End threshold not reached) **[Major Cycle]**
    1) image-domain CLEAN **[Minor Cycle]**
        a) select a subset of the PSF which includes the largest sidelobes
        b) find all points in the Residual Image above the highest sidelobe level as a fraction of the peak flux f_{max} in the image
        c) run Högbom's CLEAN on these components down to the sidelobe level and store in a minor cycle Model

    2) minor cycle Model is convolved with the PSF (via multiplication and FFTs) and subtracted from the Residual Image
    3) major cycle Model is updated by adding in the minor cycle Model

D) Convolve the major cycle Model with the 'restored' beam, and add it to the final Residual Image to produce the Restored Image

Output: Model, Residual Image, Restored Image

Cotton-Schwab's Algorithm (Ungridded-visibilities CLEAN):

Inputs: Visibilites
Parameters: gain (γ), end threshold

A) generate the PSF and Dirty Image
B) Make a copy of Dirty Image, call it the Residual Image
C) Create an empty Model

D) while(End threshold not reached) **[Major Cycle]**
    1) image-domain CLEAN **[Minor Cycle]**
        a) select a subset of the PSF which includes the largest sidelobes
        b) find all points in the Residual Image above the highest sidelobe level as a fraction of the peak flux f_{max} in the image
        c) run Högbom's CLEAN on these components down to the sidelobe level and store in a minor cycle Model

    2) minor cycle Model is convolved with the PSF (via multiplication and FFTs), de-gridded, and subtracted from the visibilities.
    3) Generate a new Dirty Image
    4) major cycle Model is updated by adding in the minor cycle Model

E) Convolve the major cycle Model with the 'restored' beam, and add it to the final Residual Image to produce the Restored Image

Output: Dirty Image, PSF, Model, Residual Image, Restored Image

CLEAN Practicals

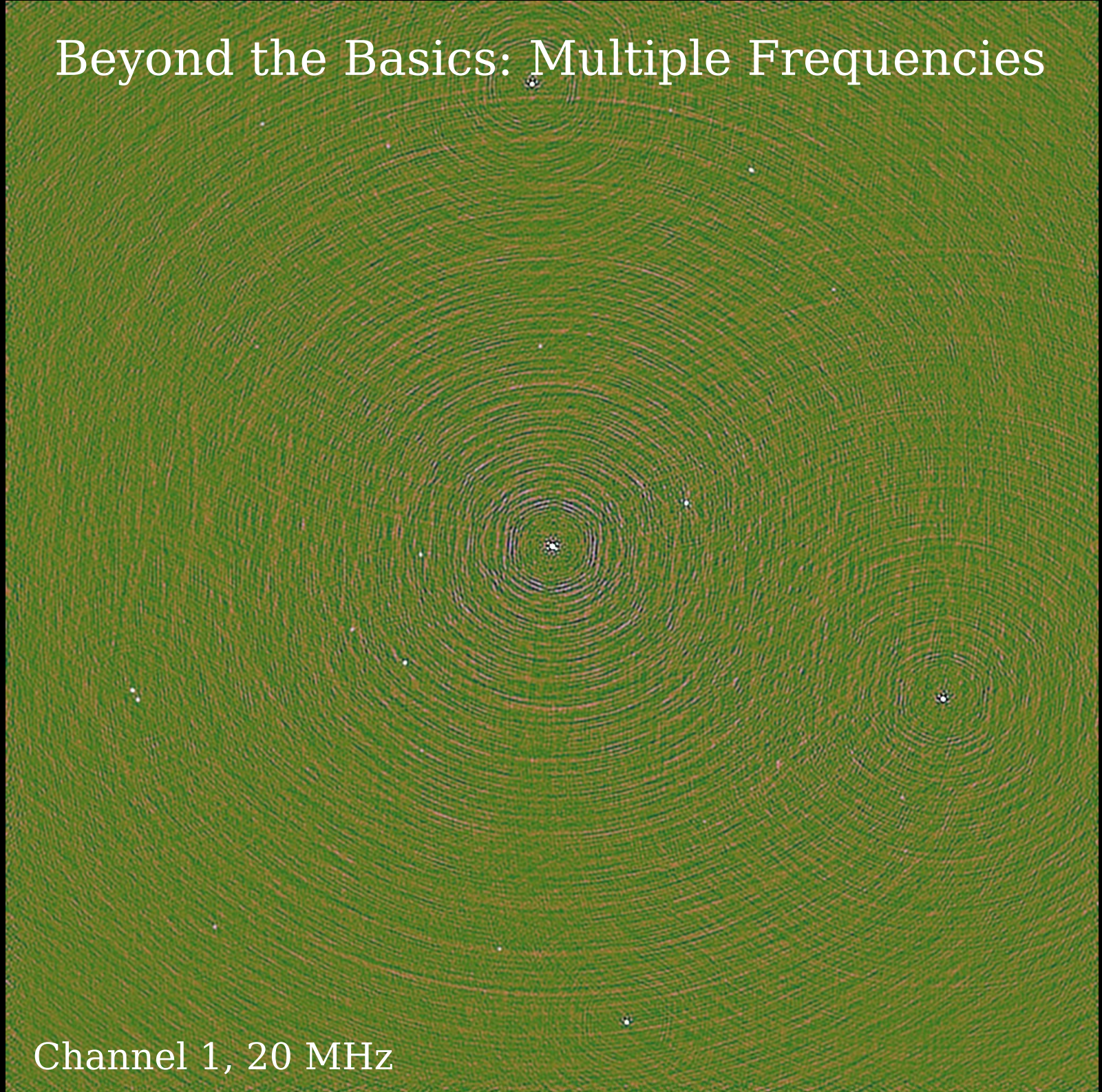
Gain: default is usually 0.1 (max: 1.0), if the gain is too low then the algorithm will take too long to converge. If it is too large, the coarseness of the deconvolution can lead to artefacts.

Threshold: CLEAN down to this threshold value, usually is set to 2-3 times the RMS of the image noise (in Janskys). If CLEANing based on a number of iterations, this can be set to 0 Janskys to stop from CLEANing negative flux.

Iterations: Number of cycles, depends on dynamic range and confusion noise of image. If CLEANing to a threshold values, make this number very large such that the threshold will be reached.

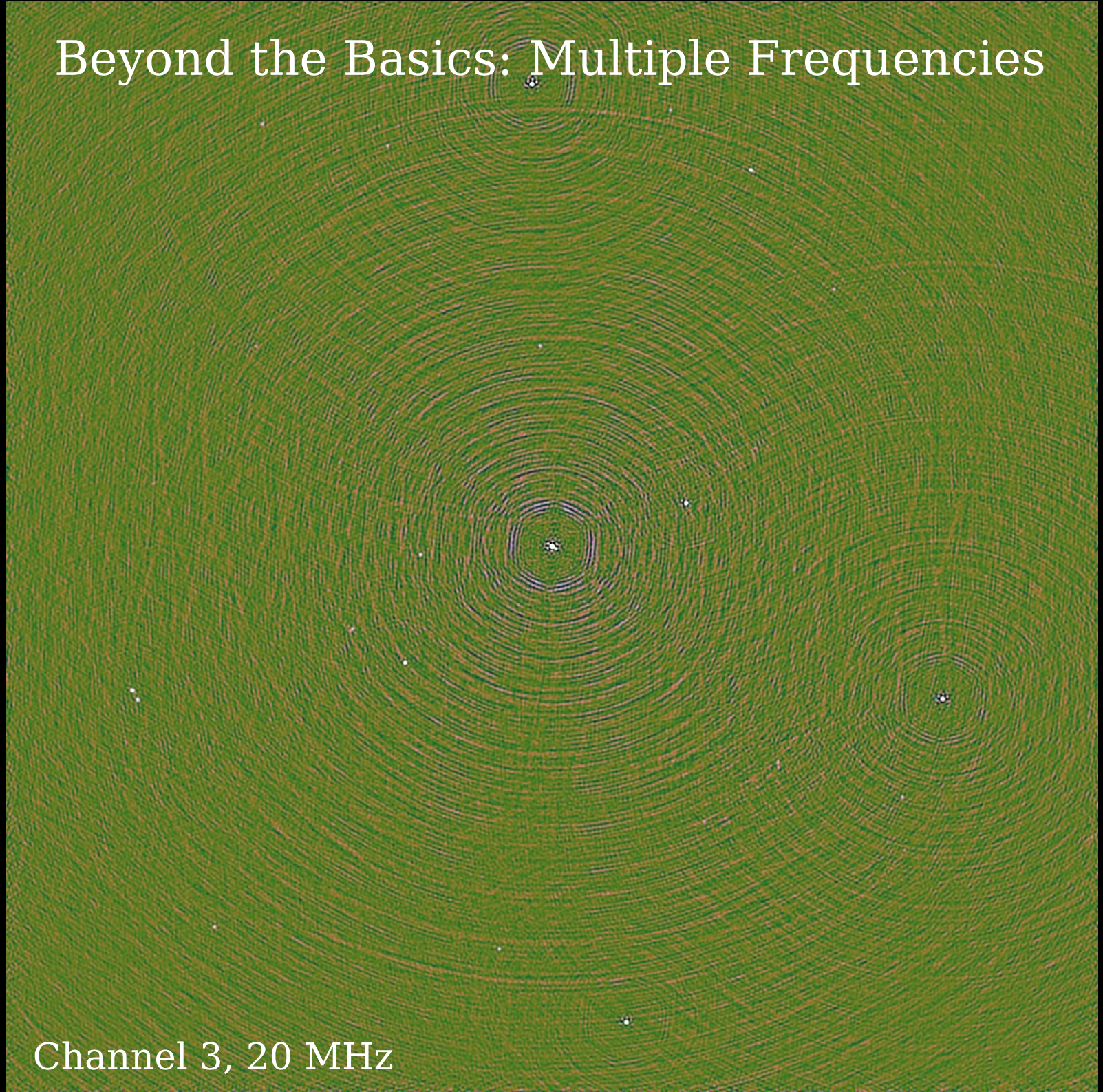
Restored Beam: A 2D Gaussian fit to the PSF to simulate a 'perfect' beam at the resolution of the array

Beyond the Basics: Multiple Frequencies



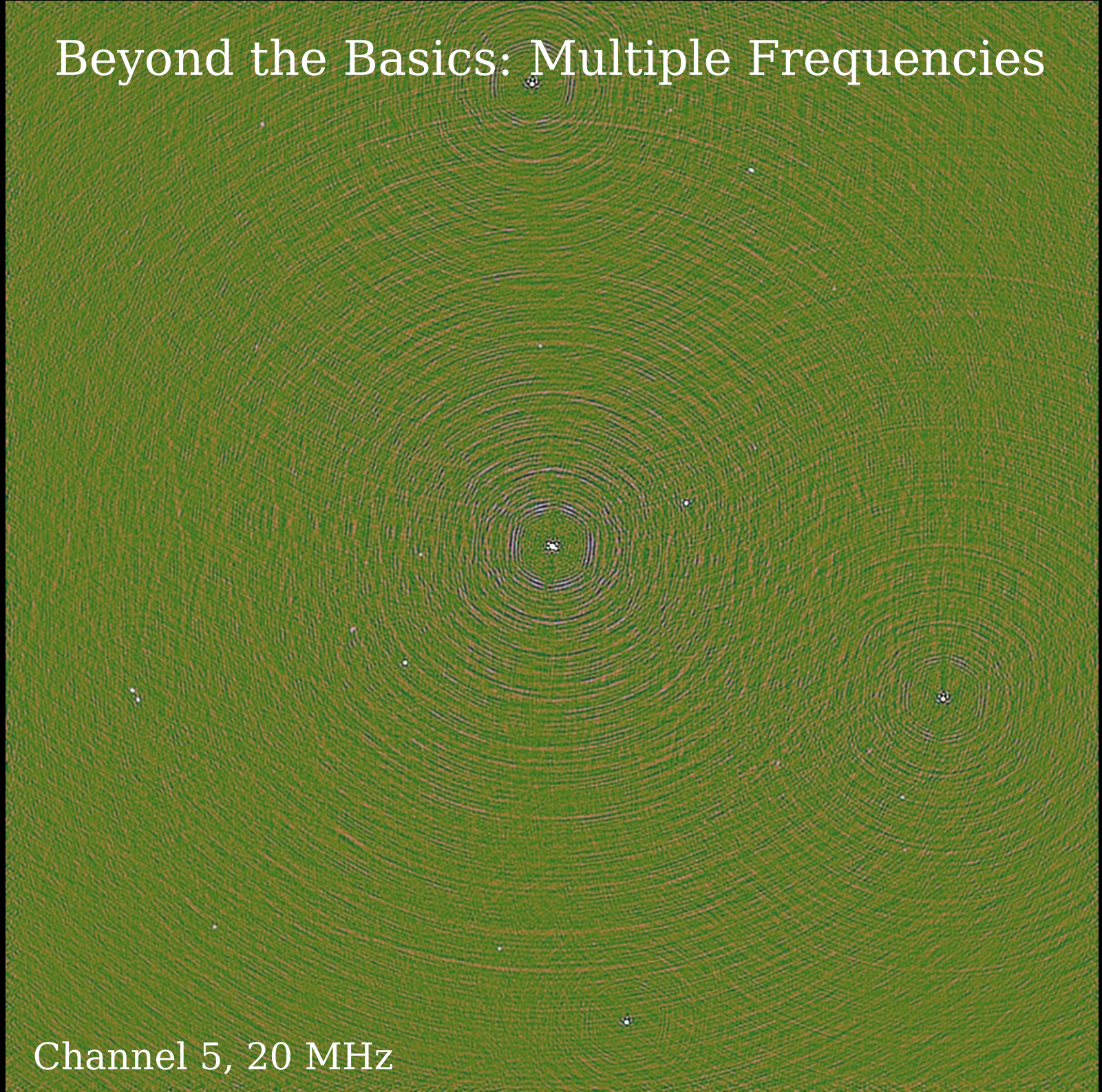
Channel 1, 20 MHz

Beyond the Basics: Multiple Frequencies



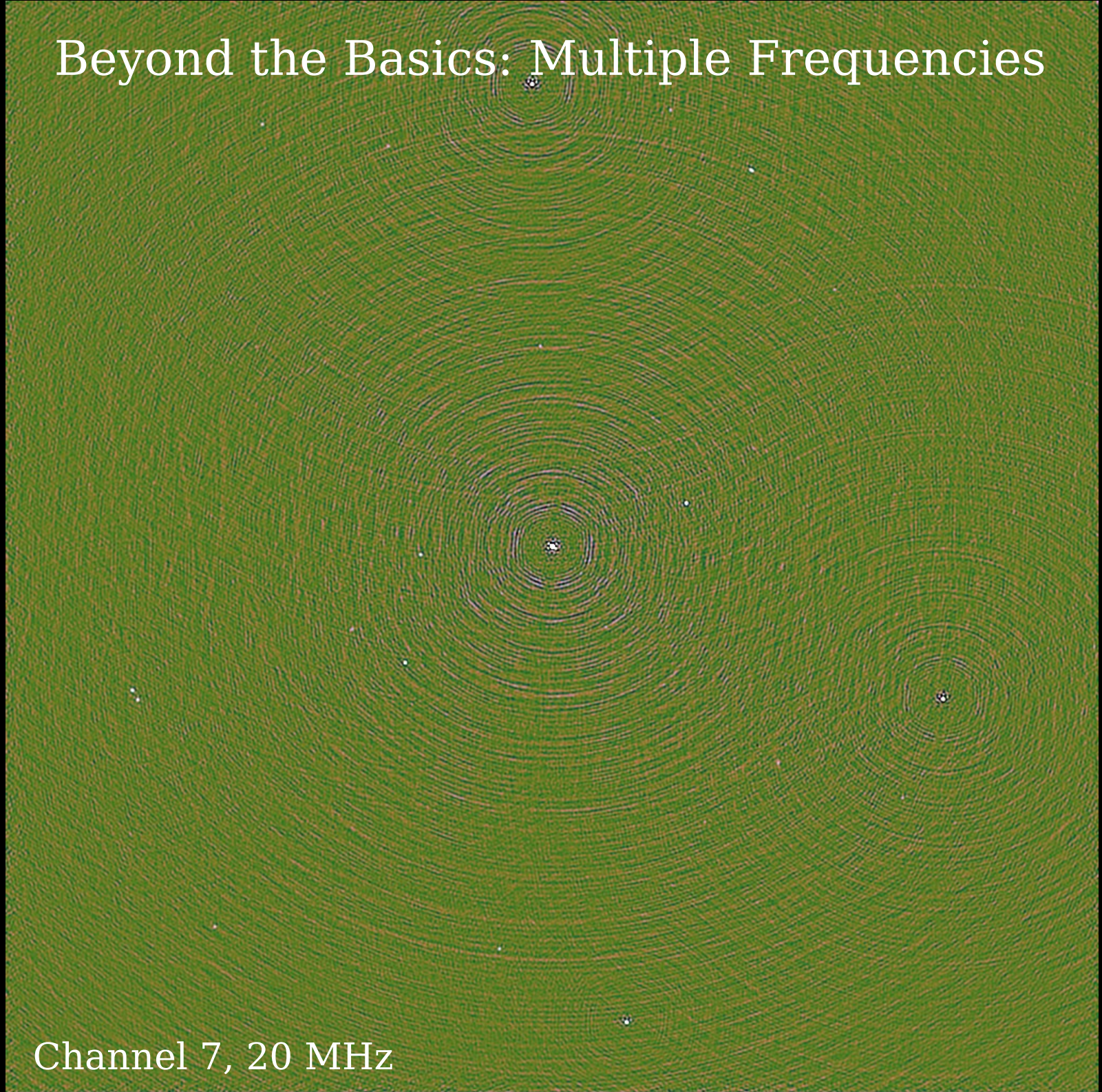
Channel 3, 20 MHz

Beyond the Basics: Multiple Frequencies



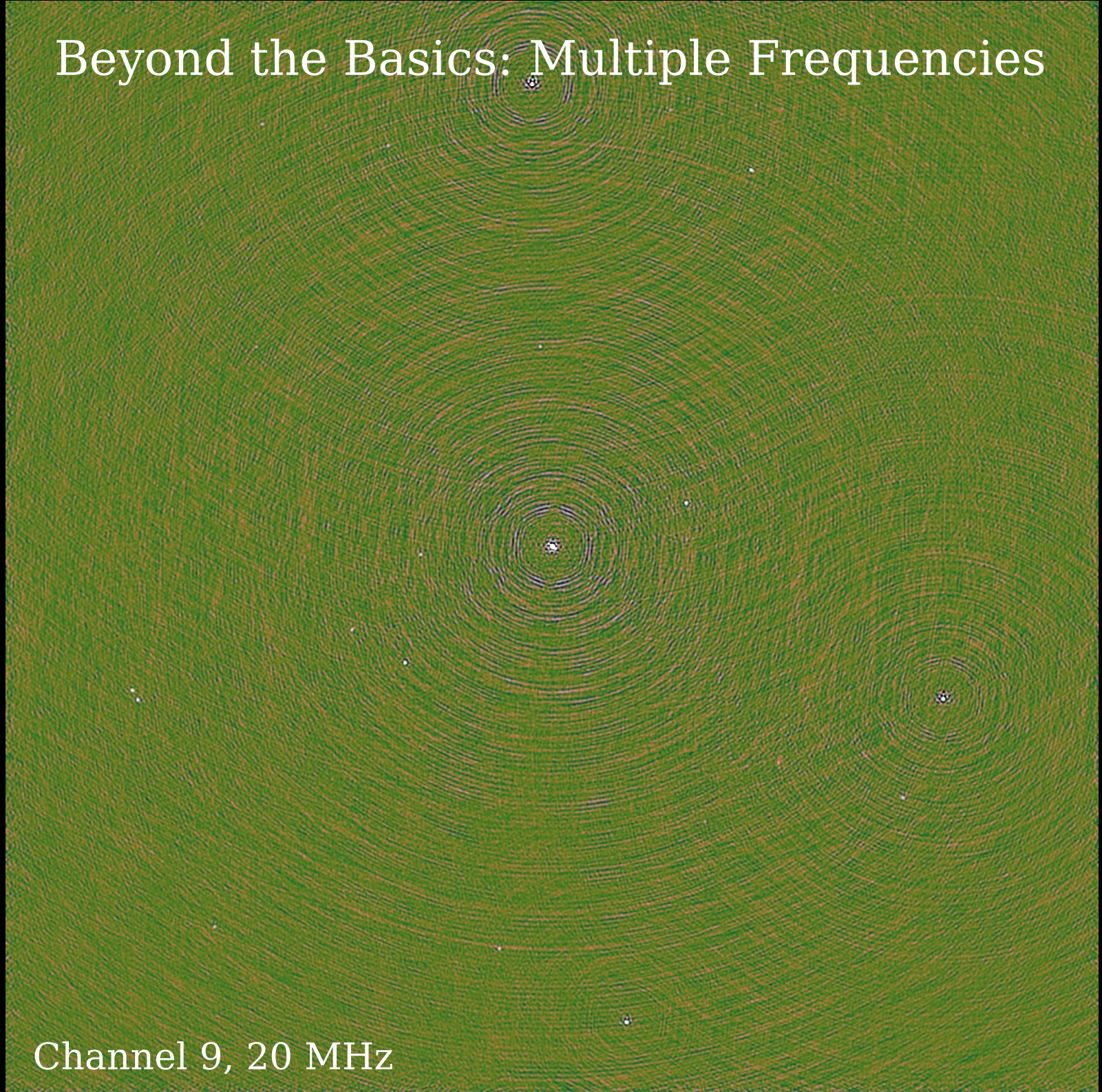
Channel 5, 20 MHz

Beyond the Basics: Multiple Frequencies



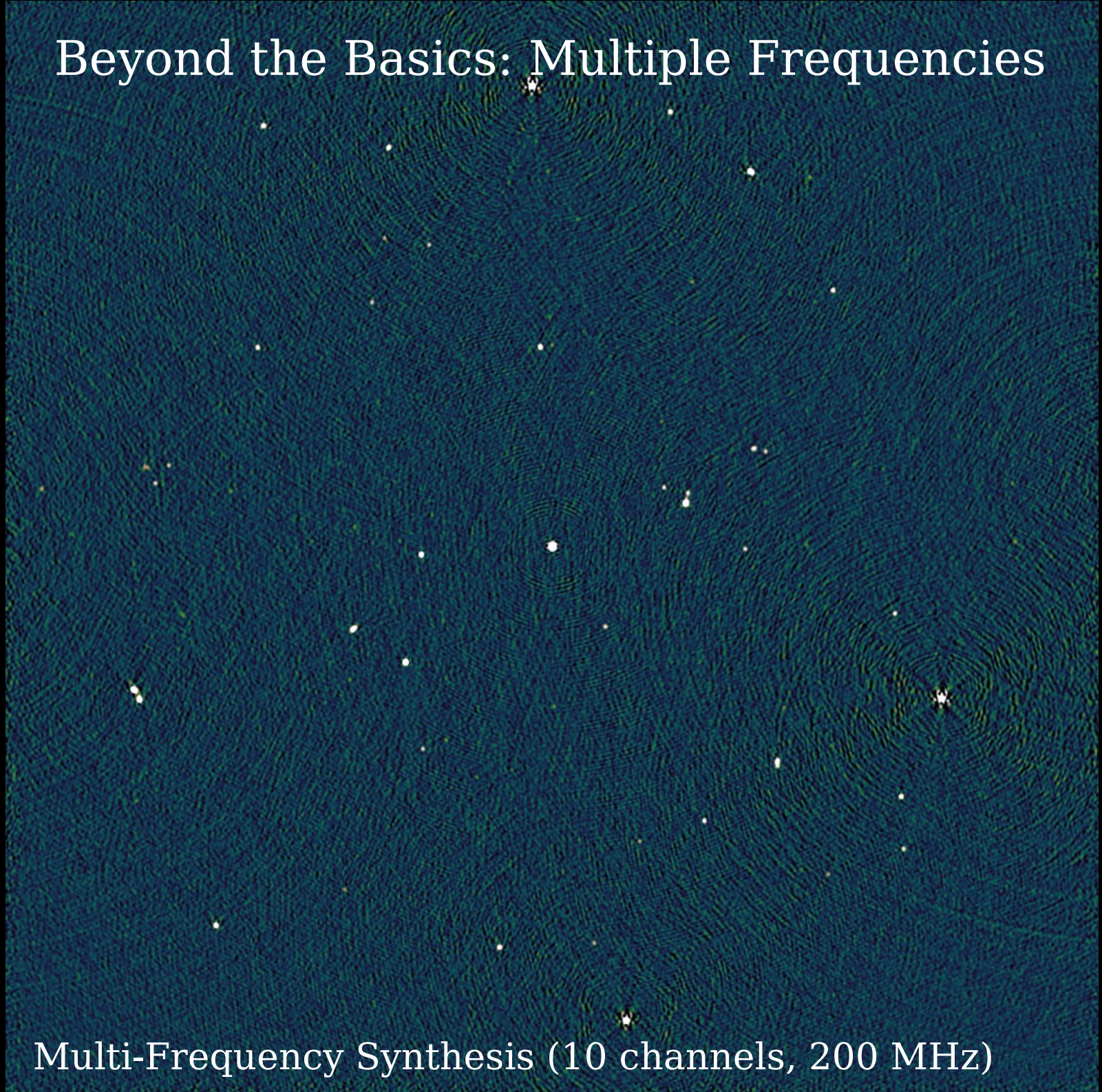
Channel 7, 20 MHz

Beyond the Basics: Multiple Frequencies



Channel 9, 20 MHz

Beyond the Basics: Multiple Frequencies



Multi-Frequency Synthesis (10 channels, 200 MHz)

Beyond the Basics: The W-term

By using a 2D Fourier Transform to create the image, we are making an assumption that the field of view is small, and thus the sky can be approximated as flat, that is the w-term is 0.

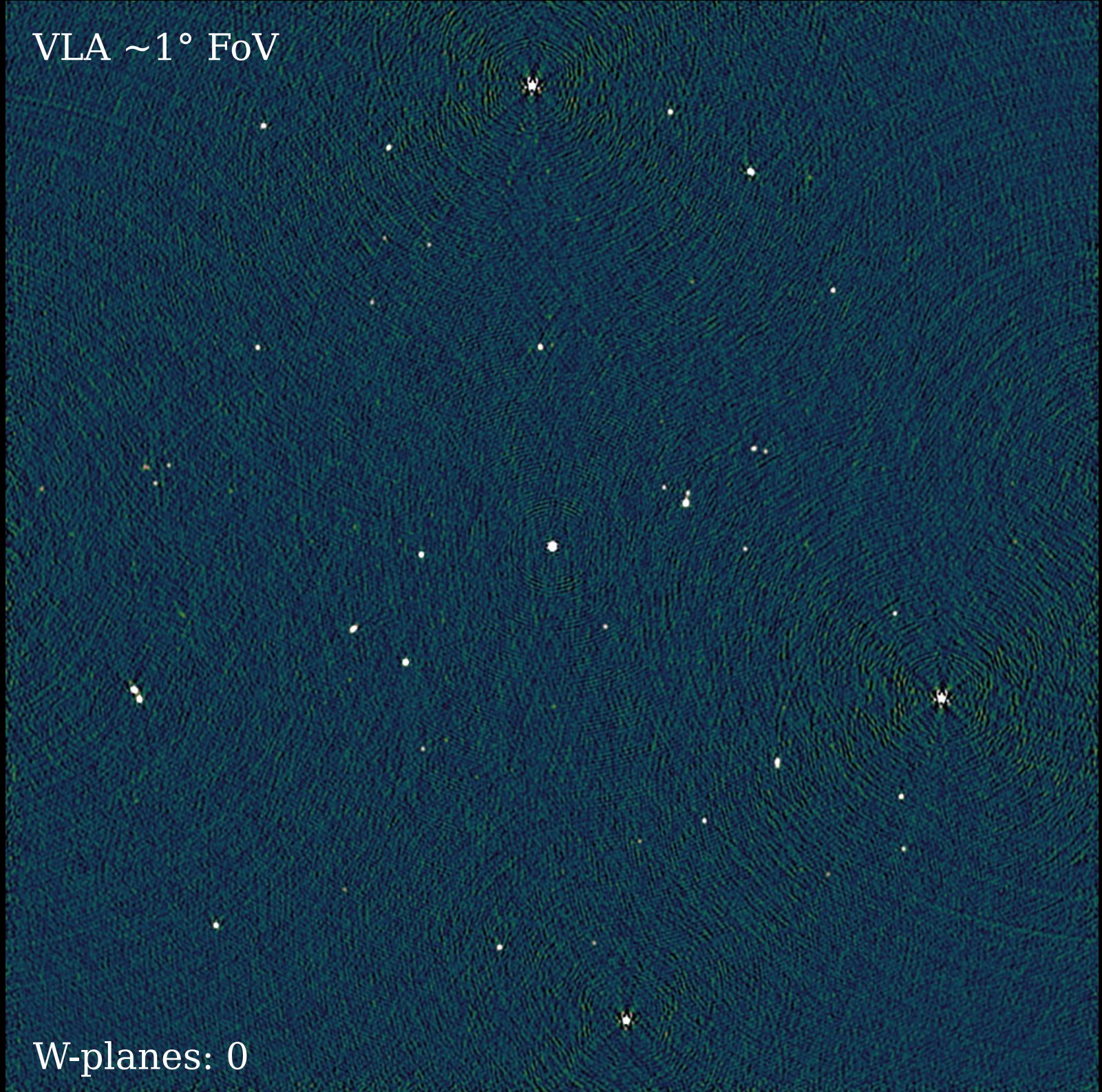
But, many of the images we make have a wide field of view, and the flat sky approximation breaks down (the w-term is not 0). There are a few methods to deal with a 3D sky.

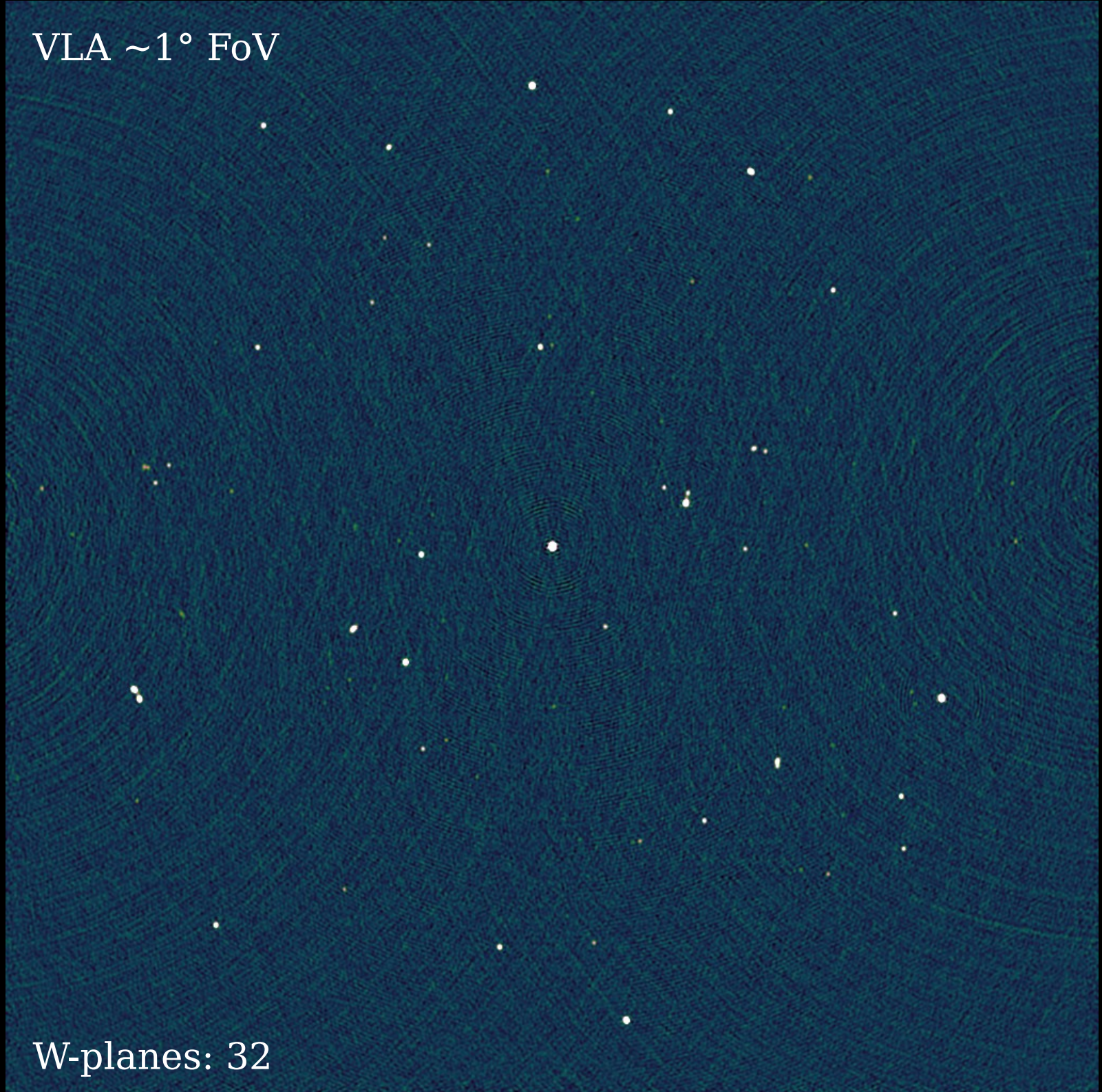
3D Fourier Transform: instead of a 2D transform, do a full 3D transform, this has a large gridding cost

Facet Imaging: Image small sub-fields, and combine the images into an image of the entire field

W-projection: convolve a kernel with the visibility data into different w-layers to approximate the w-term warping

Snapshot imaging: Image visibilities on short time-scales, and combine snapshots in the image domain.


VLA ~1° FoV
W-planes: 0


VLA ~1° FoV
W-planes: 32

Beyond the Basics: Sources that are not Points

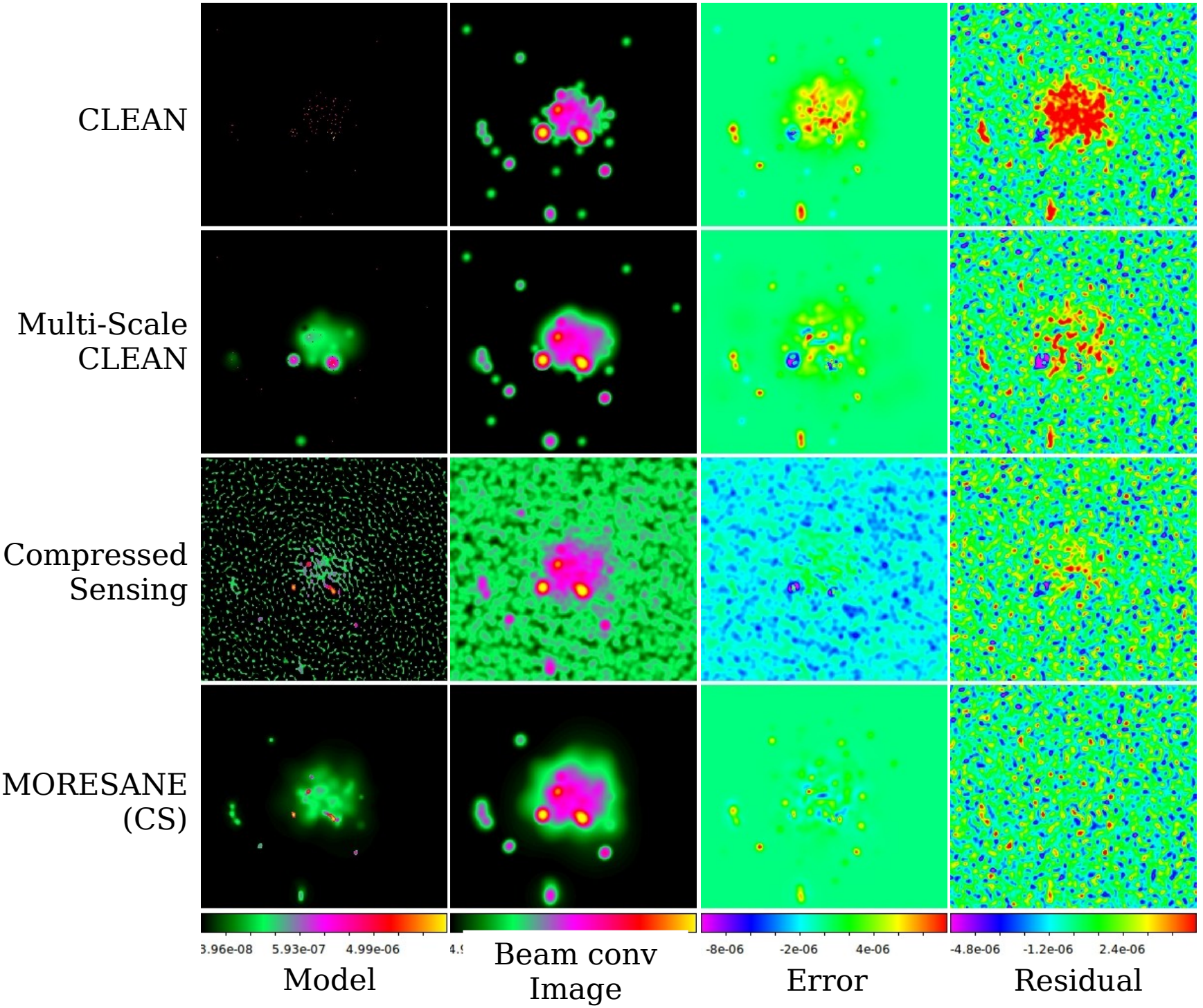
The classic CLEAN algorithms in interferometry make the assumption that everything in the sky is a compact point source. This is not a bad assumption, because most sources in the sky are unresolved. But there are many sources which are resolved (sometimes only on the longer baselines). There are a number of algorithms which specialize in this regime. This is a very active field of development.

Multi-scale CLEAN: instead of using just delta functions as a basis set for deconvolution, also use 2D Gaussians of different spatial scales.

Maximum Entropy Method: a spatially smooth solution is found which fits the data based on a log-loss function. This is an old one, and rarely used.

Compressed Sensing: Fit a set of basis functions (wavelets, spherical harmonics, etc.) instead of a delta function basis set to the dirty image and PSF. A lot of active development in this one.

Beyond the Basics: Sources that are not Points



Dabbech et al 2014

Imagers:

- CASA imager: **clean** task in NRAO CASA
- lwimger: light-weight imager, part of MeqTrees distribution
- wsclean: widefield imager, http://sourceforge.net/projects/wsclean/
- mtimager: GPU-based imager, https://github.com/ska-sa/mt-imager
- pyMORESANE: https://github.com/ratt-ru/PyMORESANE

Viewers:

- tigger: FITS and LSM viewer, part of MeqTrees distribution
- kvis: FITS viewer, part of Karma distribution, http://www.atnf.csiro.au/computing/software/karma/
- ds9: FITS and HEALPIX map viewer, http://ds9.si.edu/site/Home.html