Indirect Imaging in the SKA ERA

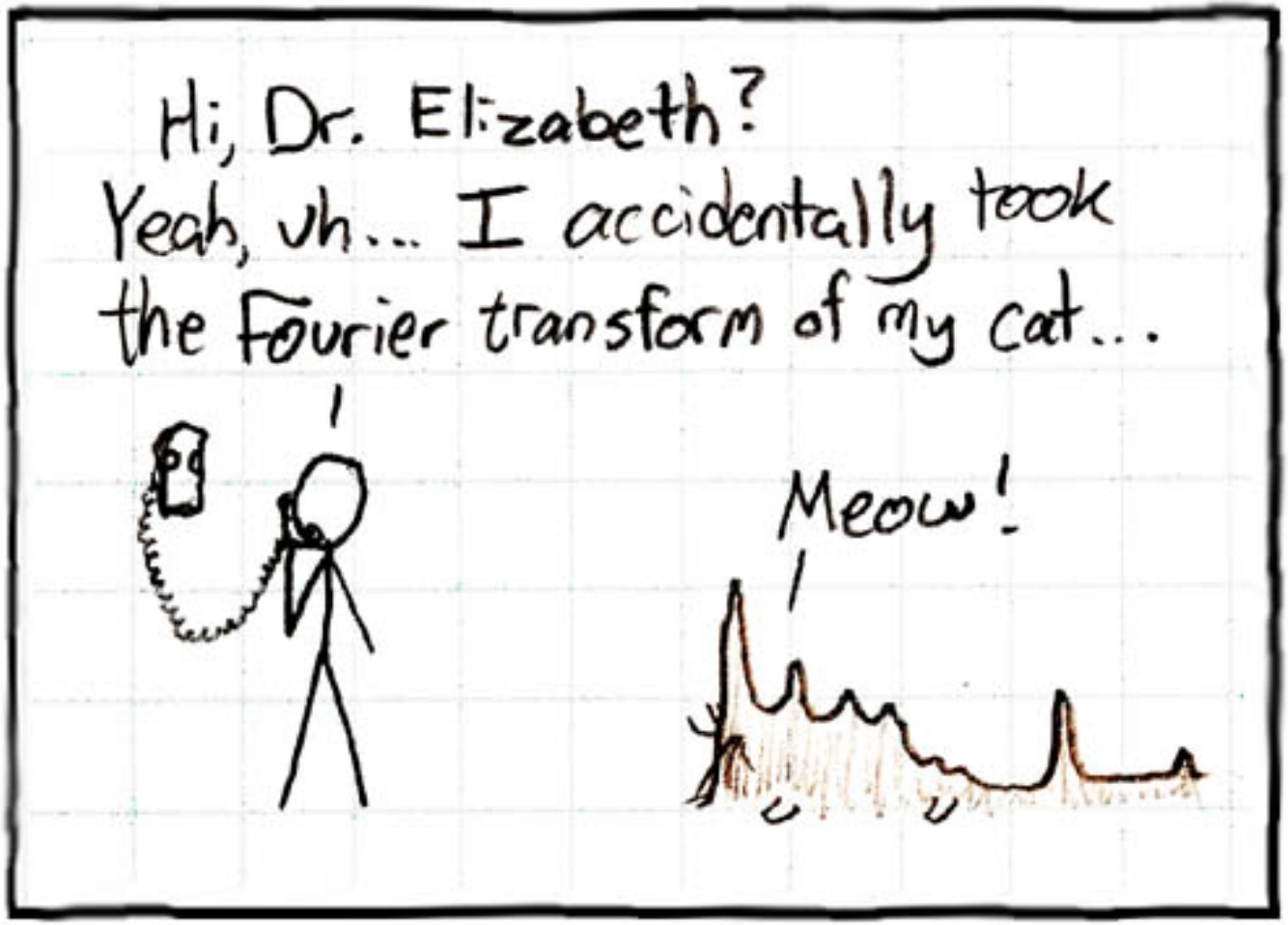
Anna Scaife

Jodrell Bank Centre for Astrophysics
University of Manchester


@radastrat

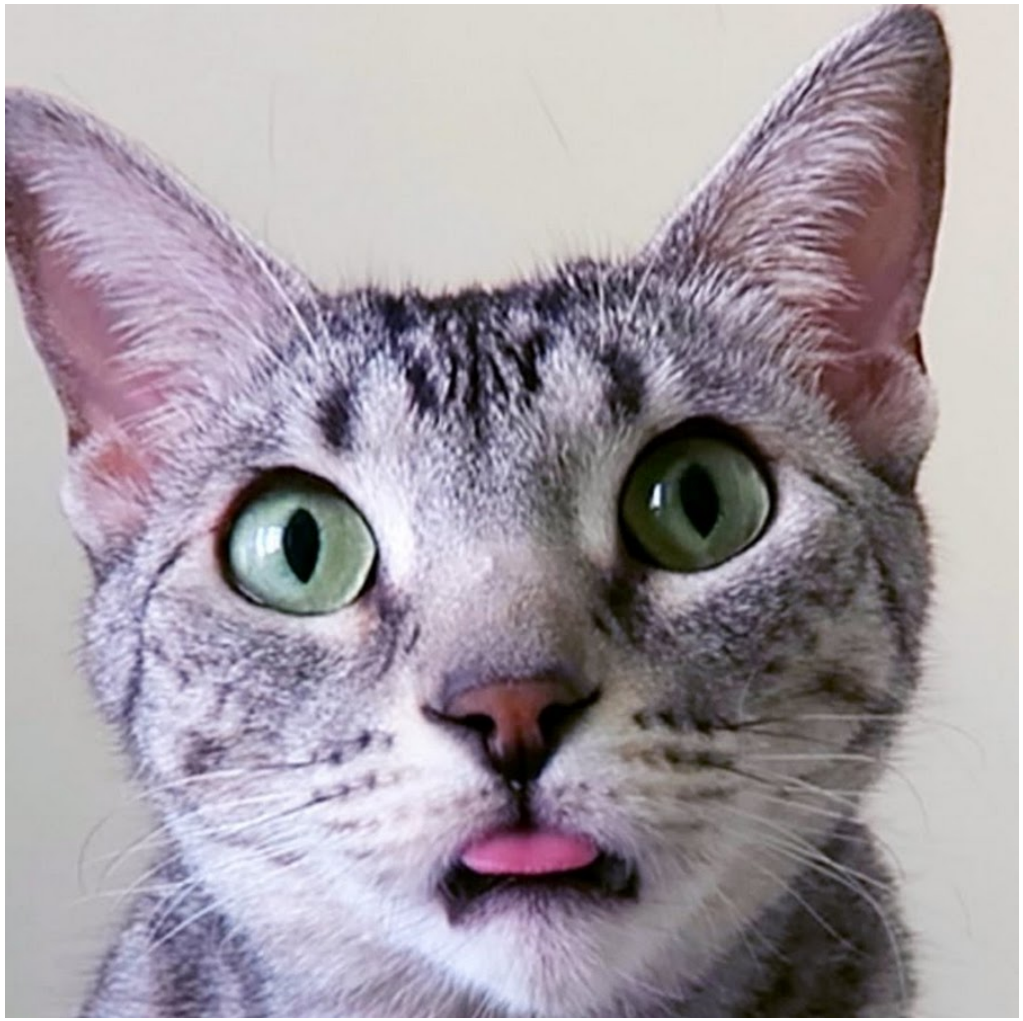
January 29, 2019

Fourier Transforms



https://imgs.xkcd.com/comics/fourier.jpg

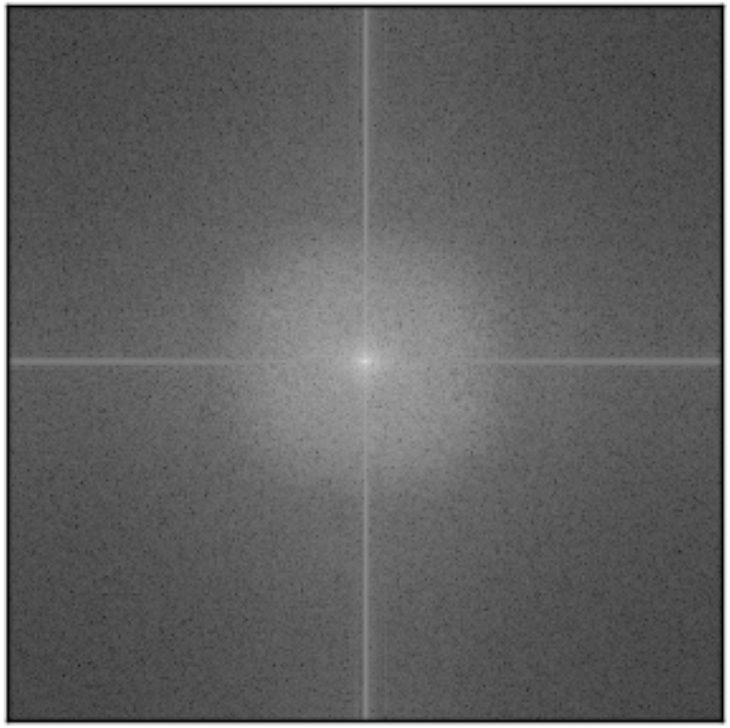
Cat Picture



Disclaimer: This is not my cat.

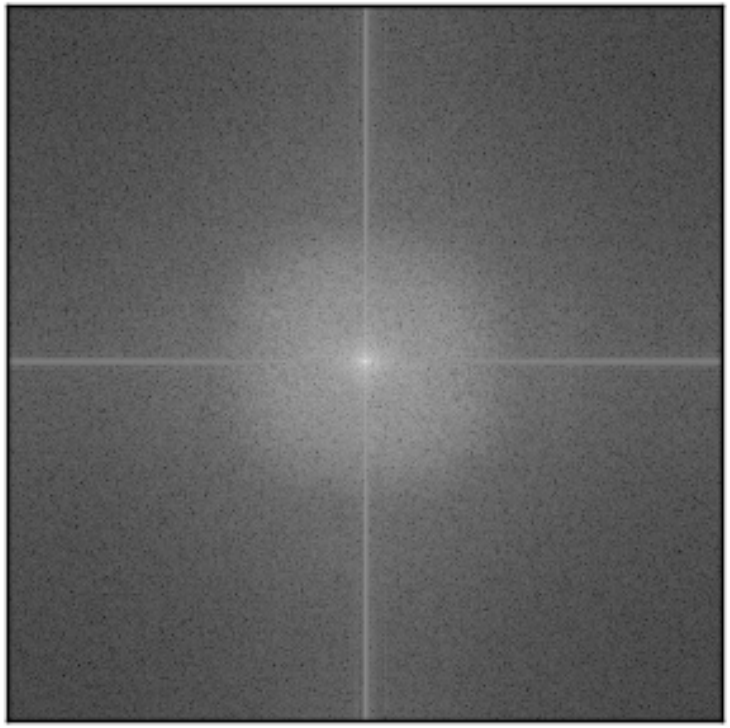
Cat

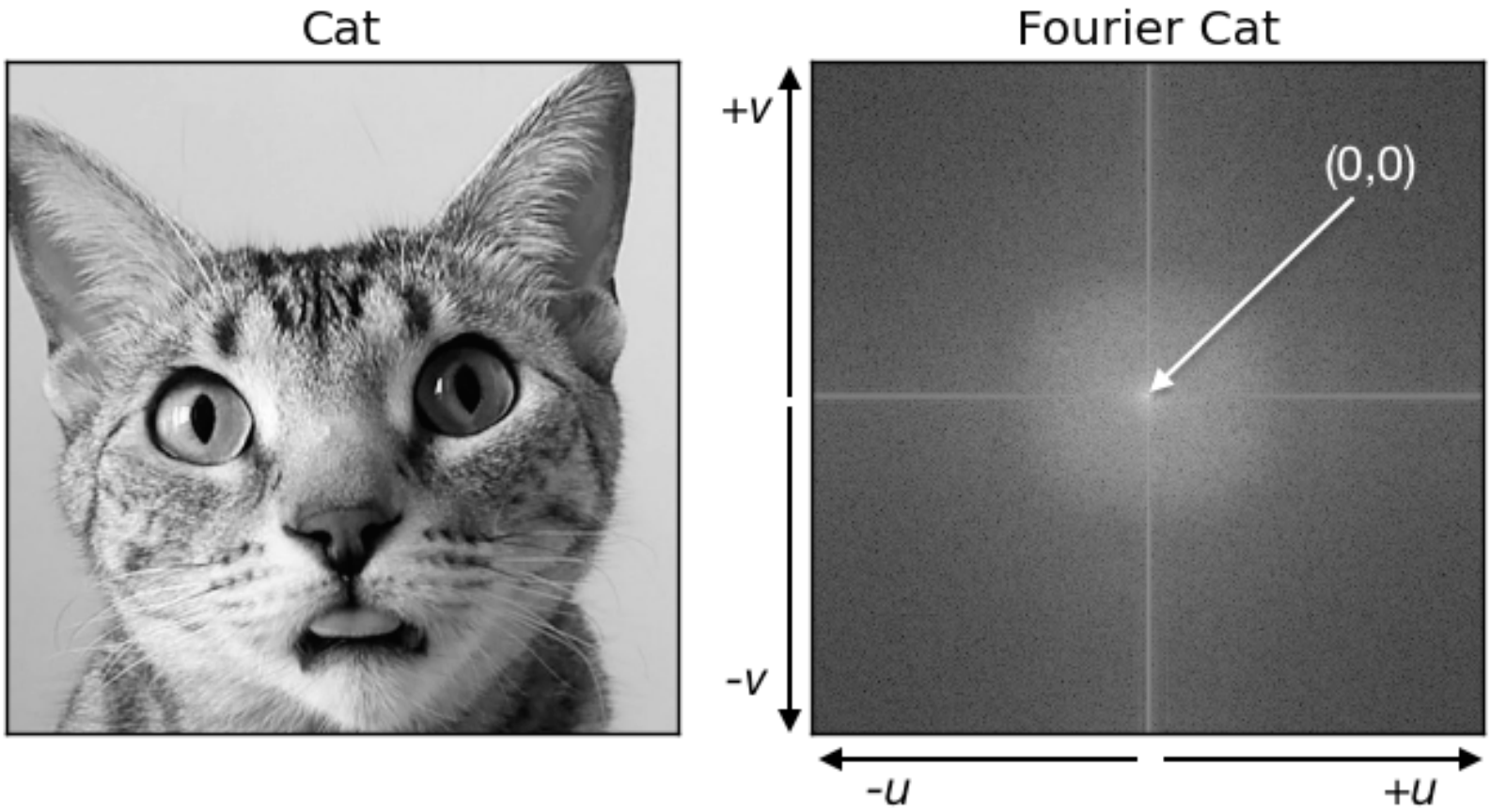
Fourier Cat



Cat

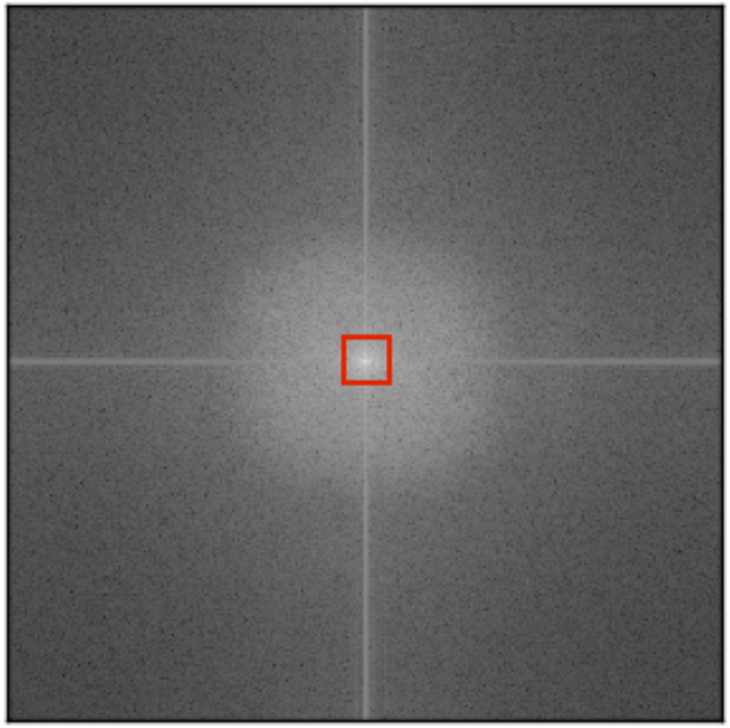
Fourier Cat



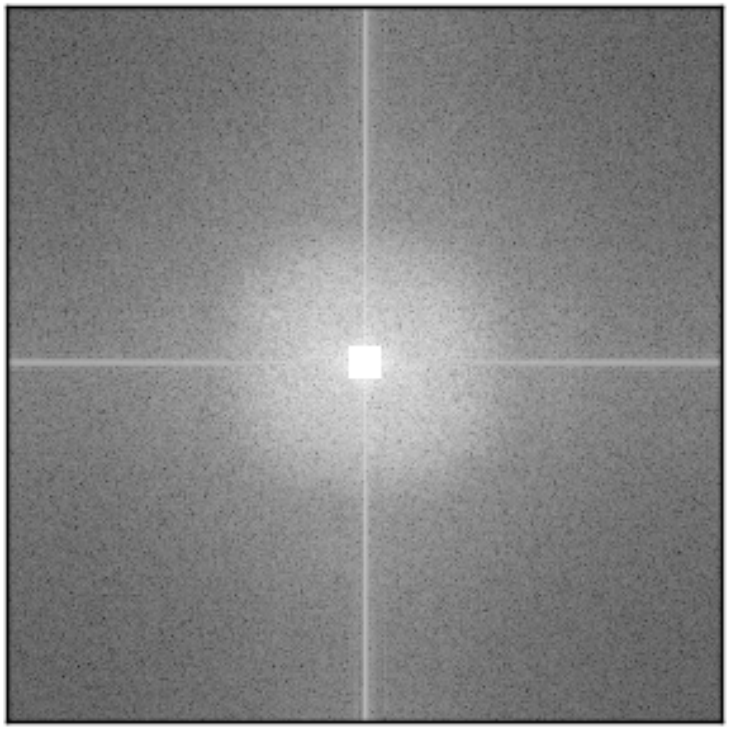

Cat
Fourier Cat
+v
-v
(0,0)
-u
+u

Cat

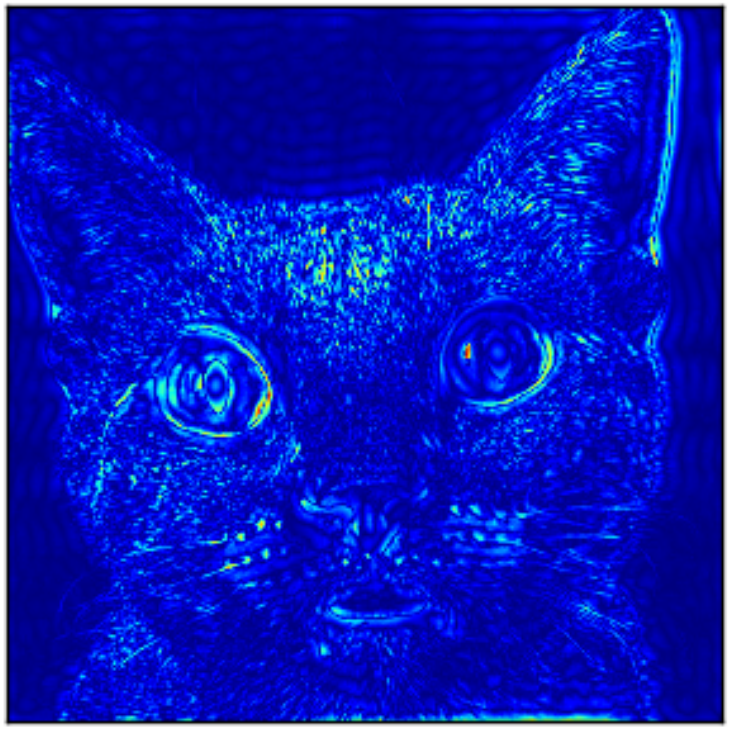
Fourier Cat



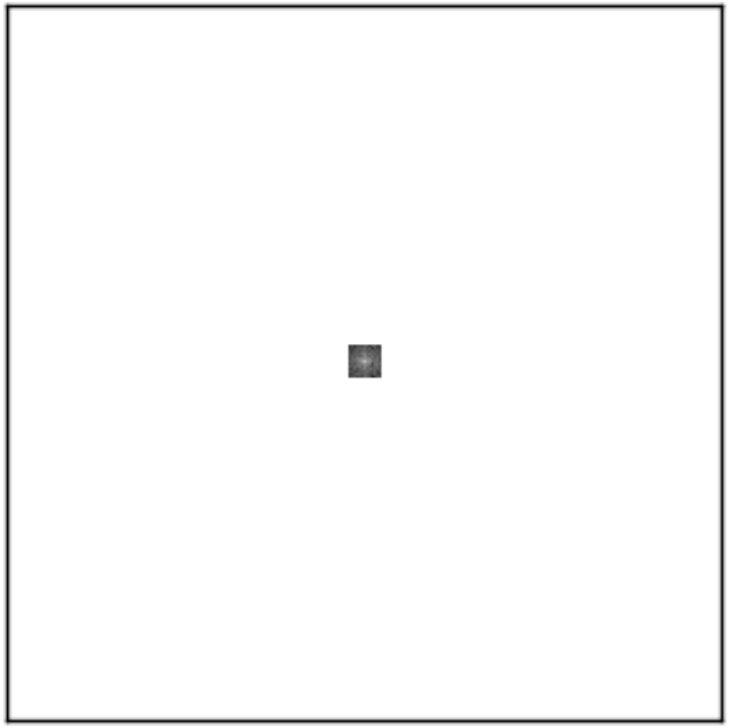
Filtered Fourier Cat



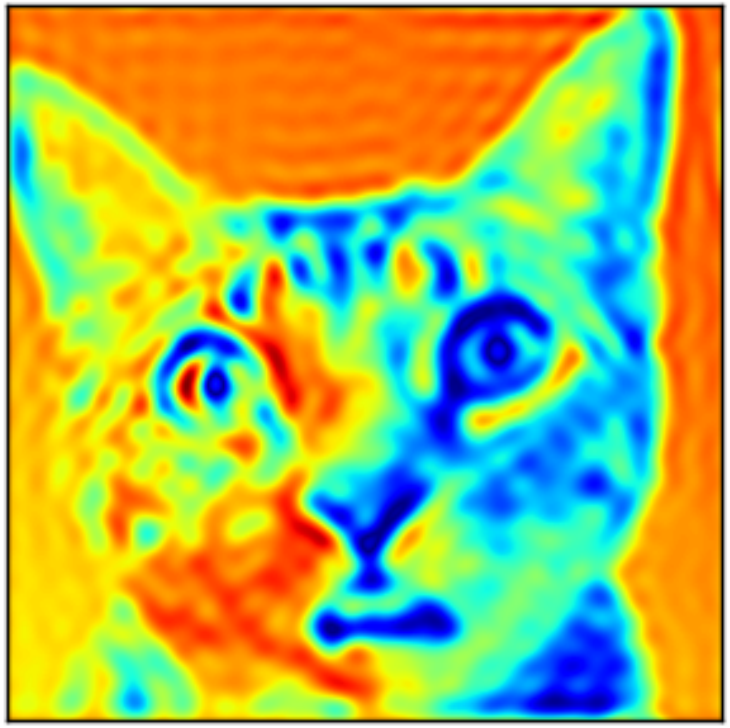
HPF Cat



Filtered Fourier Cat



LPF Cat

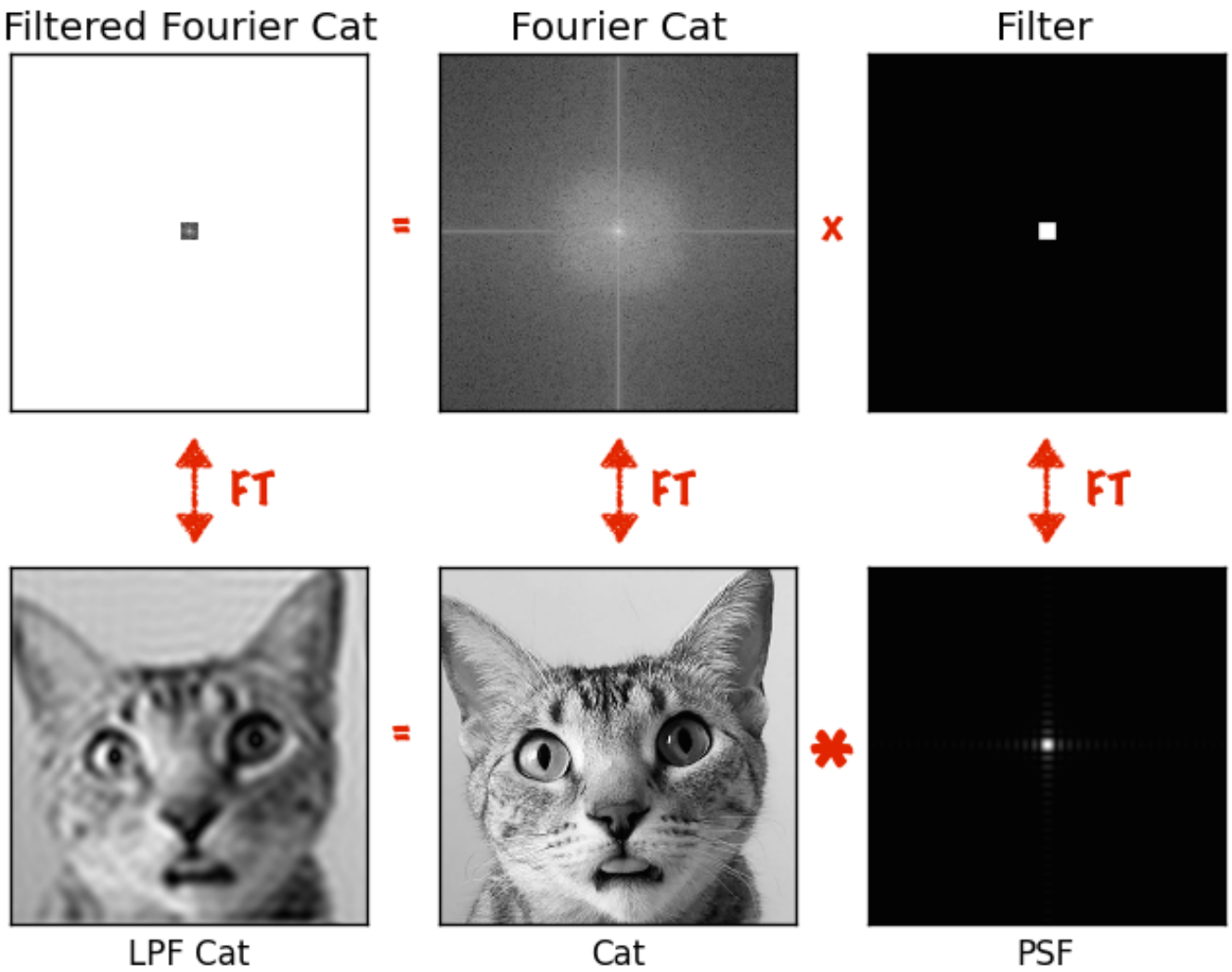


Fourier Space

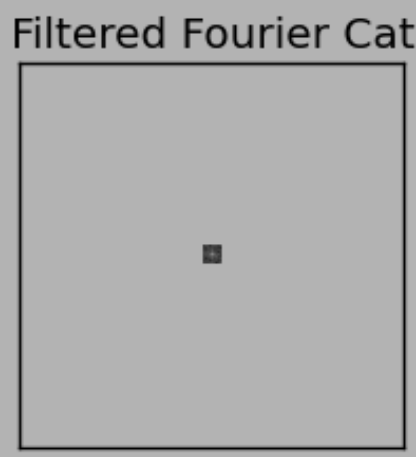
Small Fourier Frequencies = Large Scale Image Structure
Large Fourier Frequencies = Small Scale Image Structure

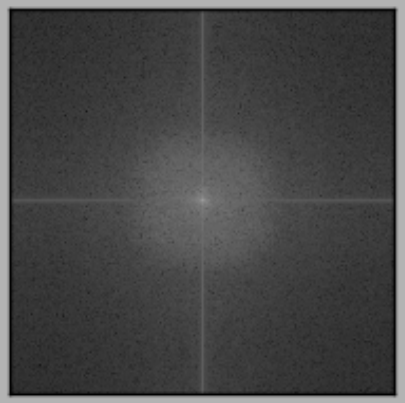
Big is Small & Small is Big

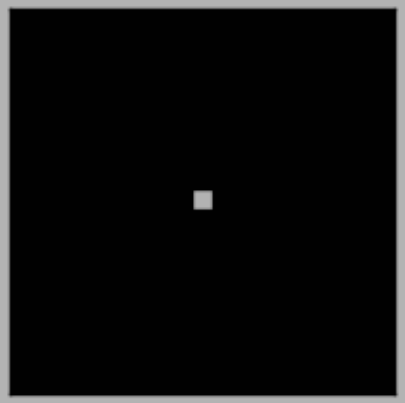
Fourier Space



Fourier Space

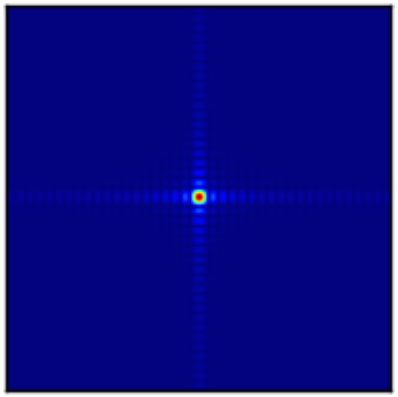







LPF Cat

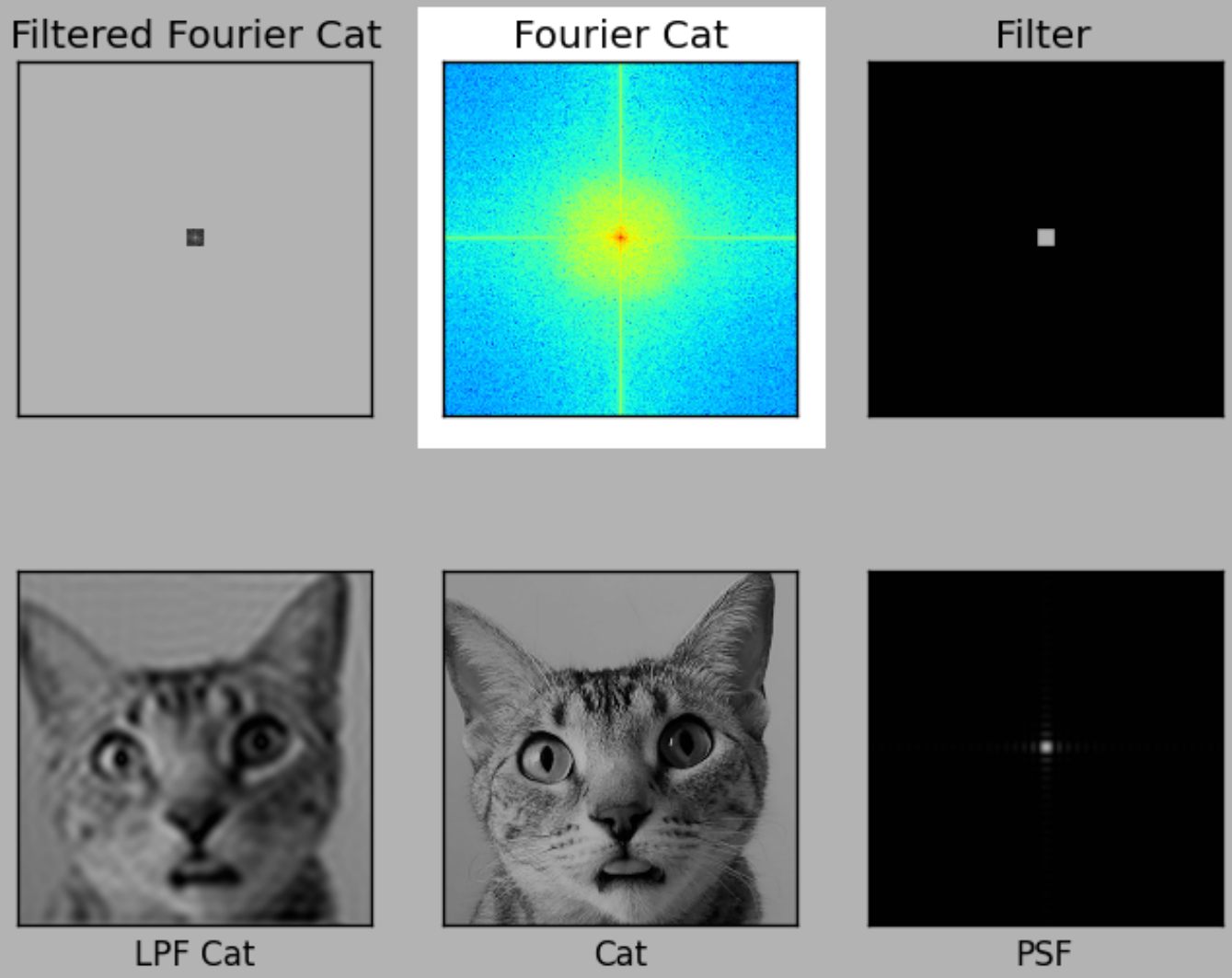

Cat


PSF

Convolution Theorem

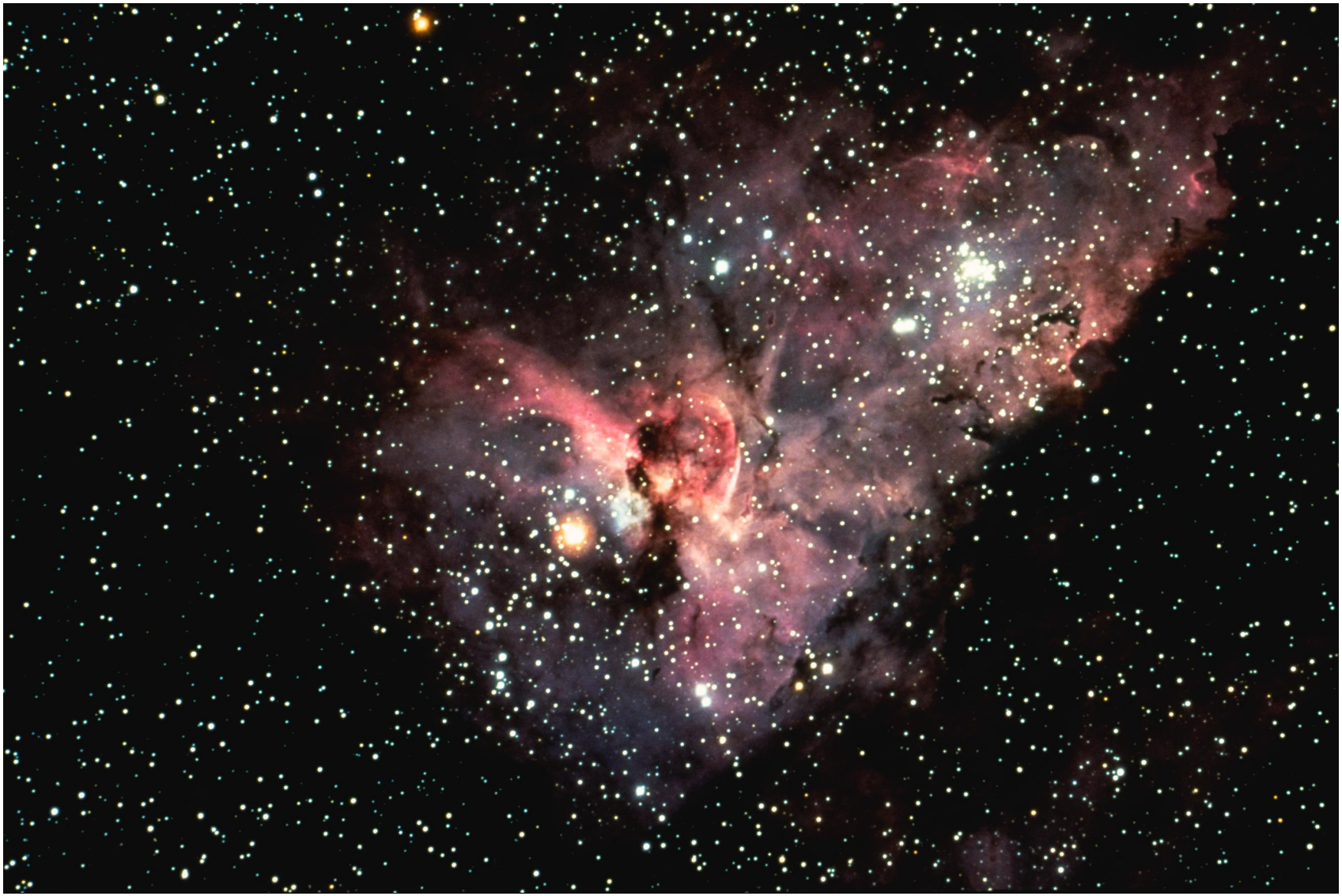
$$F(x,y) \times G(x,y) \Longleftrightarrow \tilde{F}(u,v) * \tilde{G}(u,v)$$

Fourier Space



Conjugate Symmetry

$$\tilde{F}^*(u,v) = \tilde{F}(-u,-v)$$



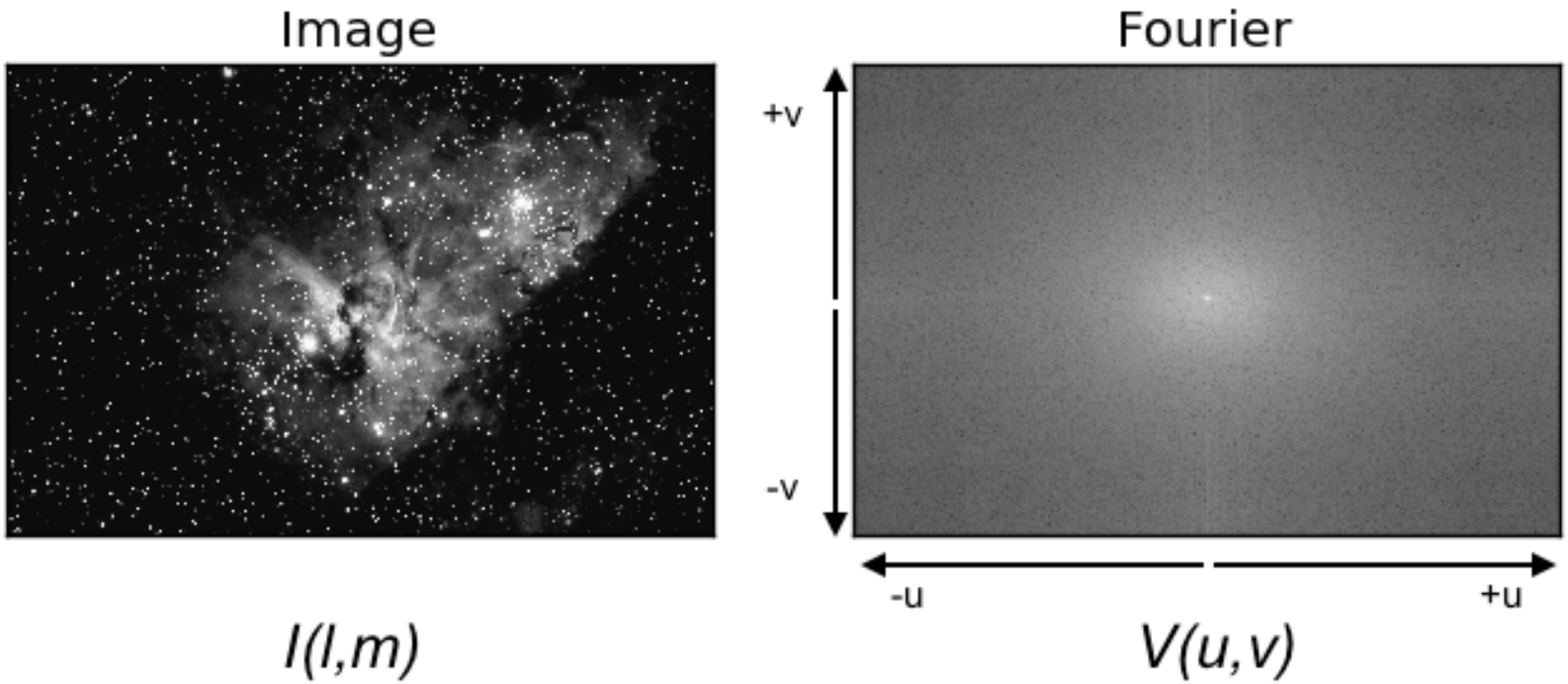
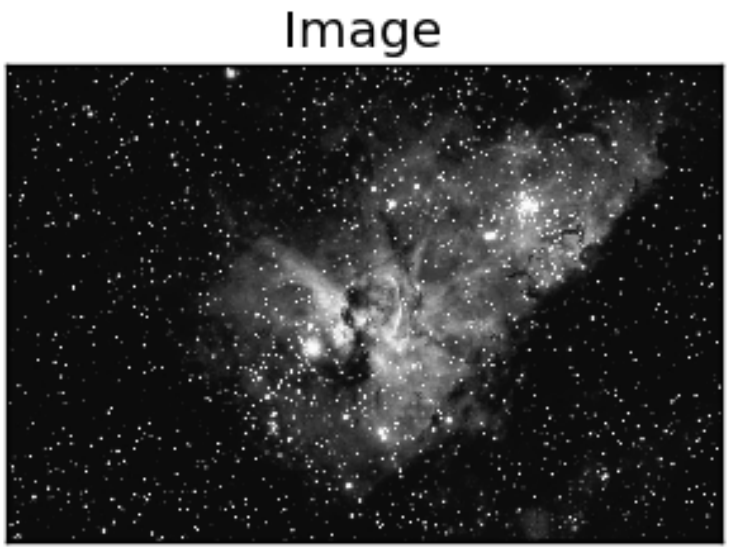
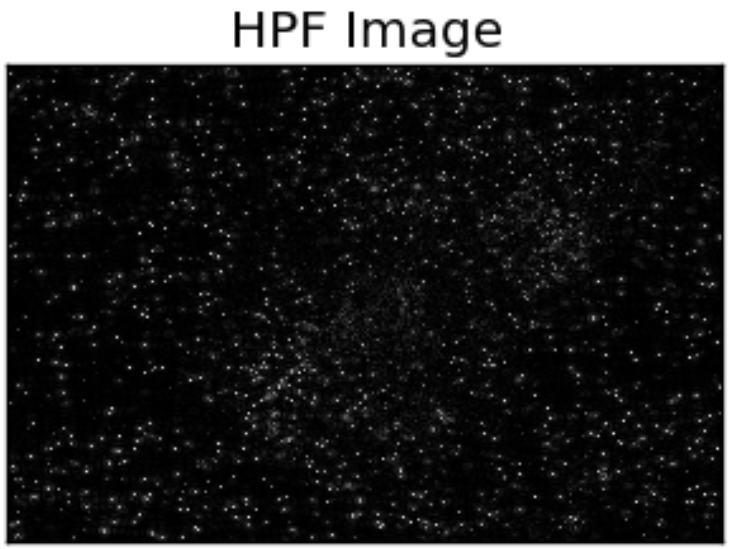


Image
Fourier
+v
-v
-u
+u
I(l,m)
V(u,v)

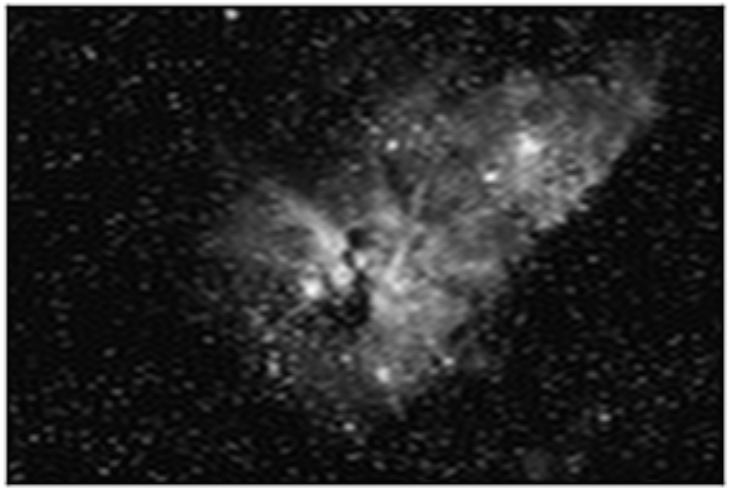


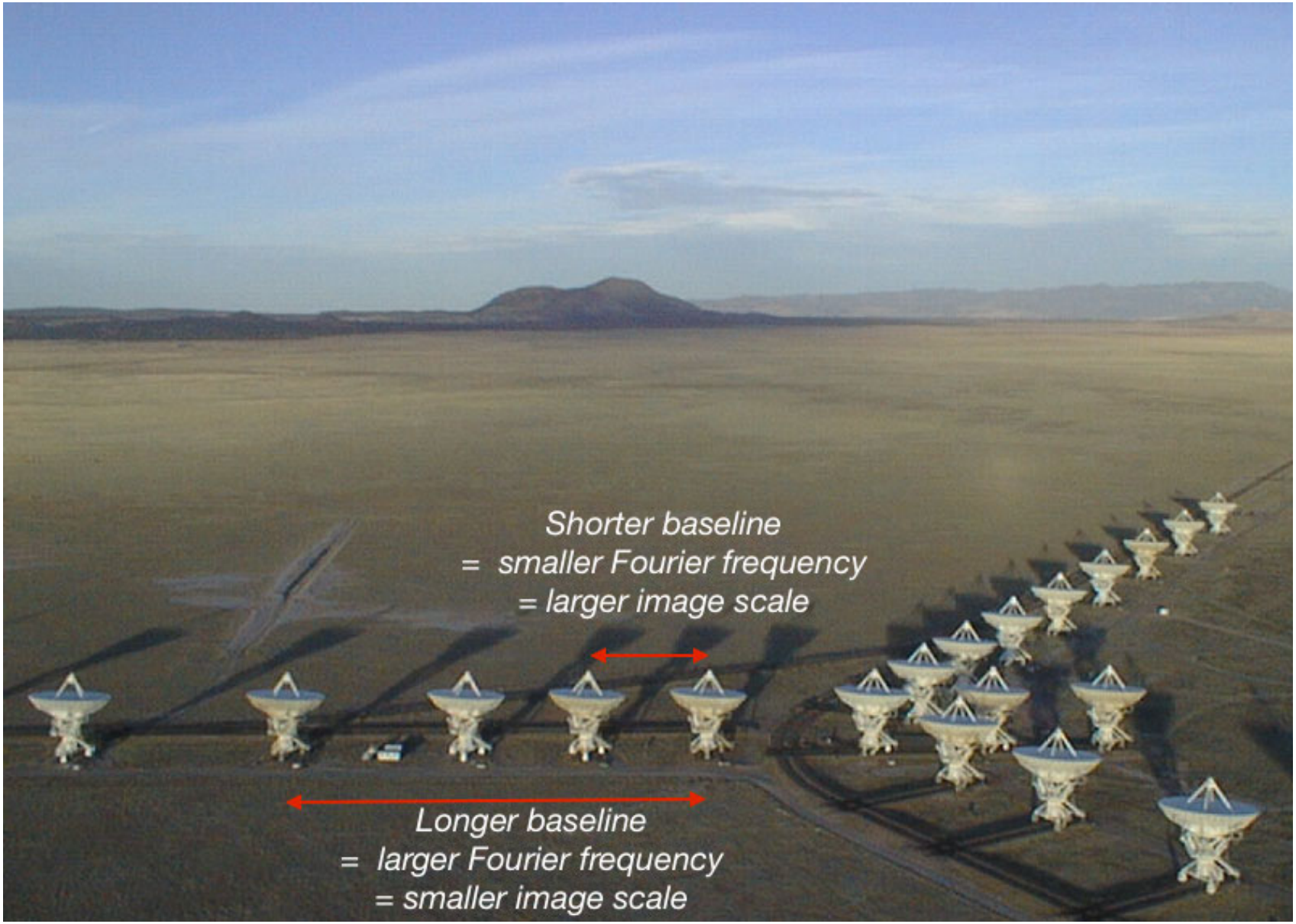
Image

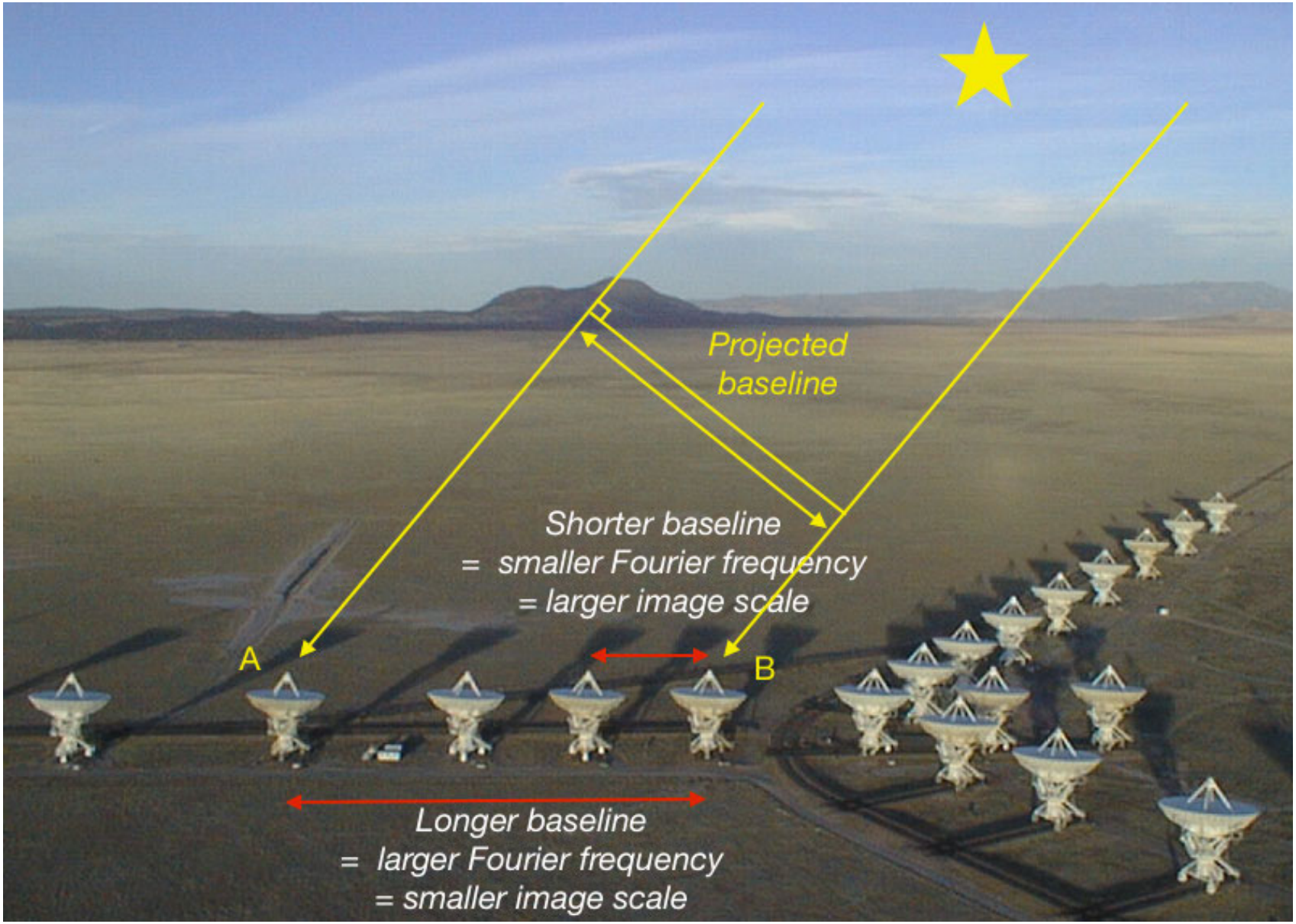

HPF Image

Image

LPF Image




Shorter baseline
= smaller Fourier frequency
= larger image scale
Longer baseline
= larger Fourier frequency
= smaller image scale


Projected
baseline
Shorter baseline
= smaller Fourier frequency
= larger image scale
A
B
Longer baseline
= larger Fourier frequency
= smaller image scale

$$I_{meas}(l,m) = \iint S(u,v)V(u,v)e^{2\pi i(ul+vm)}\,du\,dv$$

This sampling function identifies the values of (u,v) that we sample according to our baseline distribution.

$$S(u,v) = \sum_{i=1}^{M} \delta(u-u_i, v-v_i)$$

Where M is the number of different visibilities that we have:

$$M = N_{\mathrm{ant}}(N_{\mathrm{ant}}-1)/2 \times N_{\tau} \times N_{\mathrm{f}}$$

$$S(u,v) = \sum_{i=1}^{M} \delta(u-u_i, v-v_i)$$

$$\downarrow$$

$$I_{meas}(l,m) = \iint S(u,v)V(u,v)e^{2\pi i(ul+vm)}\,du\,dv$$

δ -function reminder

$$\int \delta(x-x_0)f(x)\,\mathrm{d}x = f(x_0)$$

We can use the properties of the δ -function to rewrite this integral as a sum:

$$I_{meas}(l,m) = \frac{1}{M}\sum_{i=1}^{M} V(u_i,v_i)e^{2\pi i(u_i l+v_i m)}$$

$$I_{meas}(l,m) = \frac{1}{M}\sum_{i=1}^{M} V(u_i,v_i)e^{2\pi i(u_i l + v_i m)}$$

$$= \frac{1}{M}\sum_{i=1}^{M} V(u_i,v_i)\left[\cos[2\pi(u_i l + v_i m)] + i\sin[2\pi(u_i l + v_i m)]\right]$$

This is a complex quantity, but the sky intensity is real.

$$\boxed{V(-u,-v) = V^*(u,v)}$$

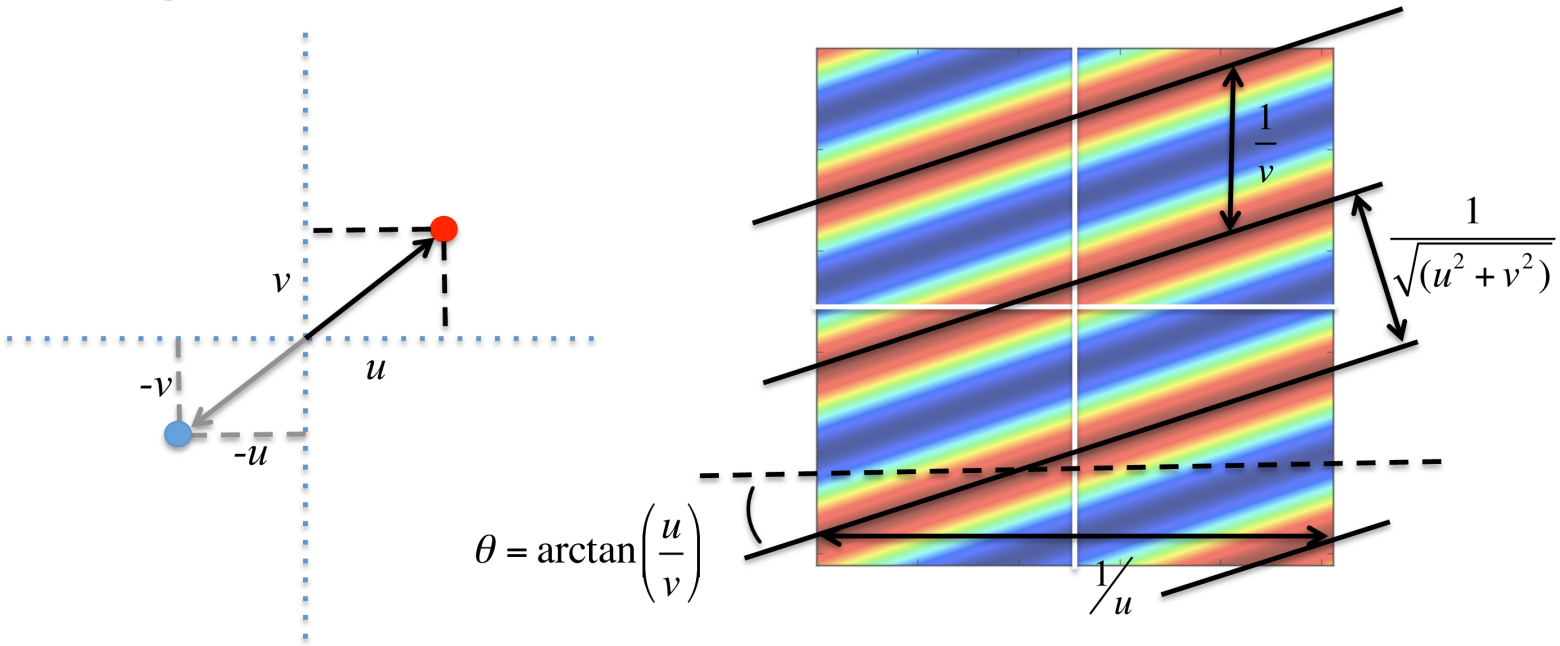
If we change our notation slightly, so that $V = Ae^{i\phi}$, we can write:

$$I_{meas}(l,m) = \frac{1}{M}\sum_{i=1}^{M} A(u_i,v_i)\cos[2\pi(u_i l + v_i m) + \phi_i]$$

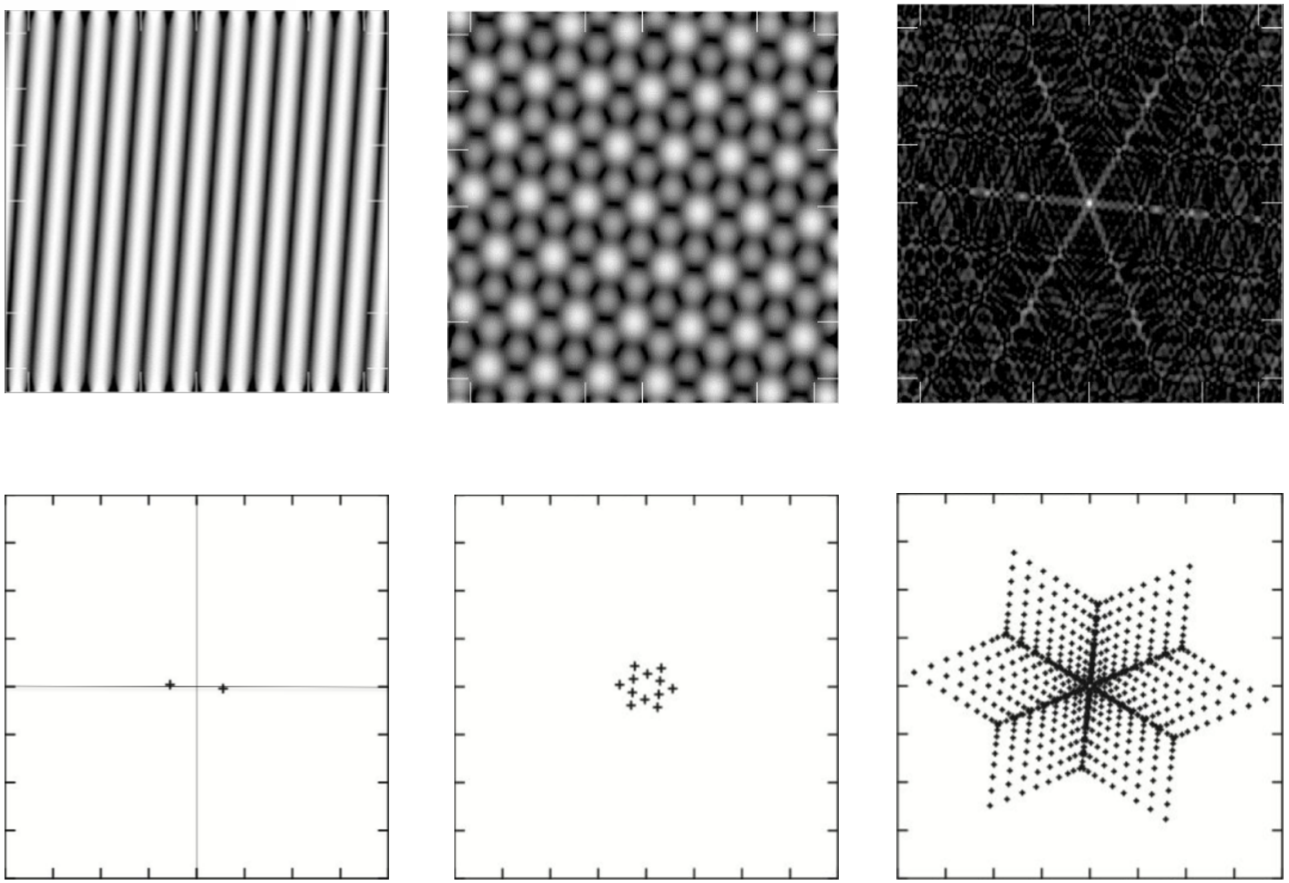
Visibility Components

$$I_{meas}(l,m) = \frac{1}{M}\sum_{i=1}^{M} A(u_i,v_i)\cos[2\pi(u_i l + v_i m) + \phi_i]$$

Writing the equation in this way allows us to visualise how our image is composed.

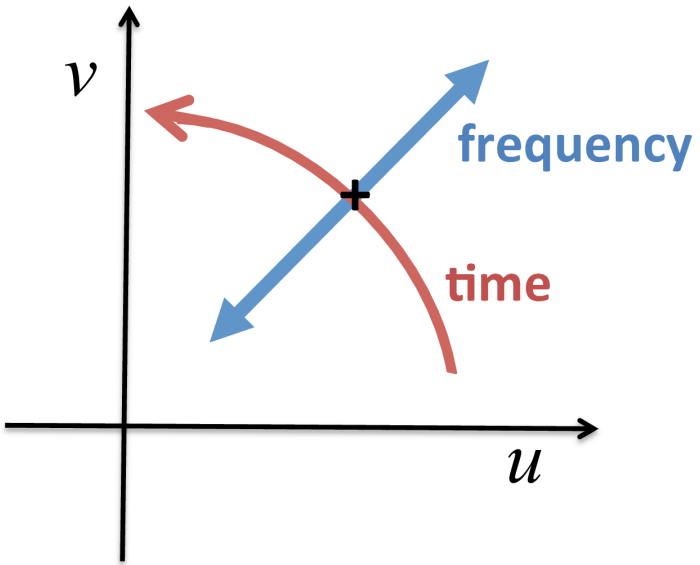


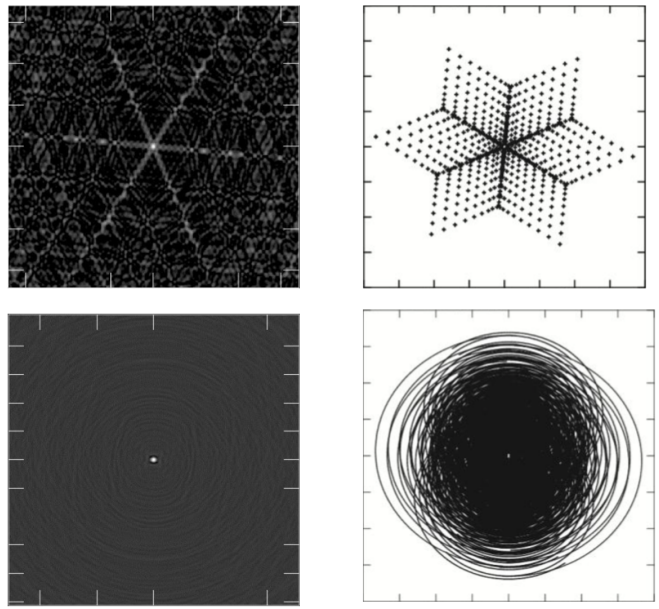
Synthesized Beam



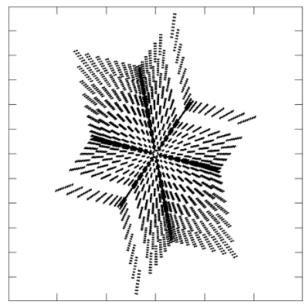
Synthesized Beam

- A change in frequency produces a radial change in (u,v).
- A change in time produces a ~azimuthal change in (u,v).



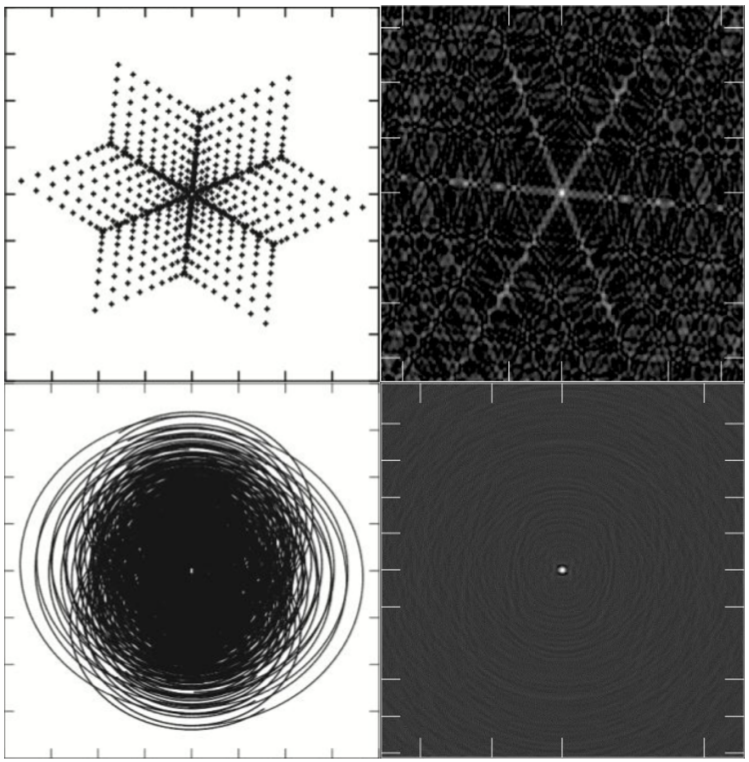


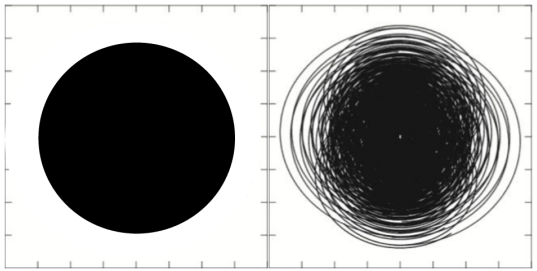
- The response from multi-frequency data is a little more complicated (later).



Synthesized Beam

$S(u,v)$ $\quad$ $\mathrm{FT}^{-1}\left[S(u,v)\right]$



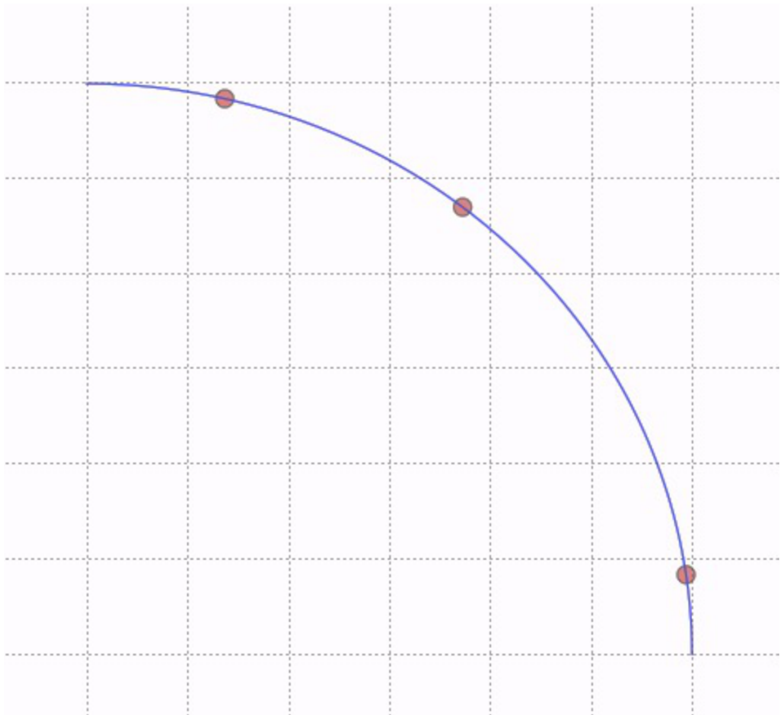


The baseline (uv) sampling defines the measured angular scales and sets the resolution.

<u>Disambiguation:</u>
Synthesized beam
= point spread function
= dirty beam

Gridding

FFTs are faster but they also introduce complications.

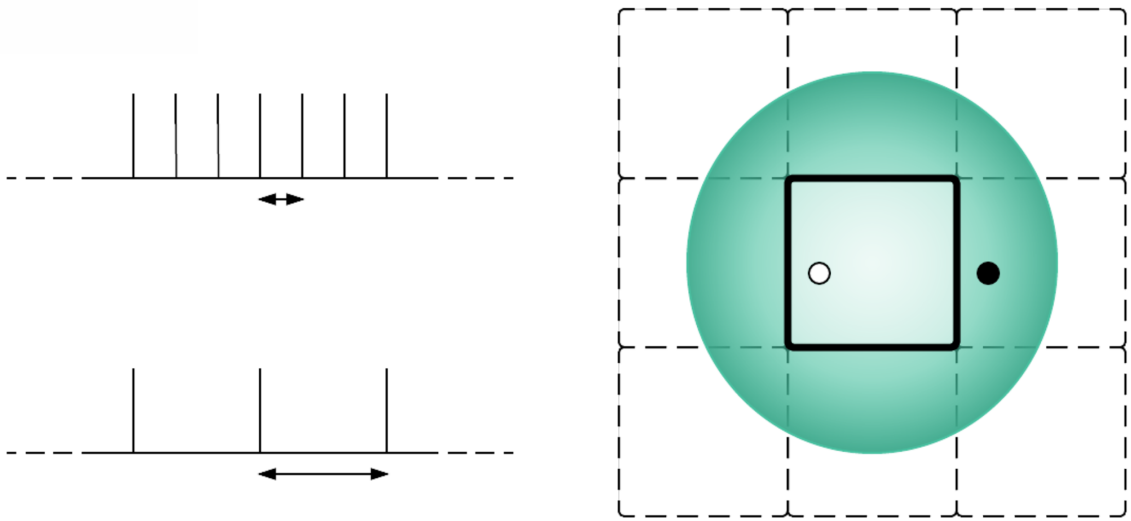


FFTs require regularly spaced (u,v) data.

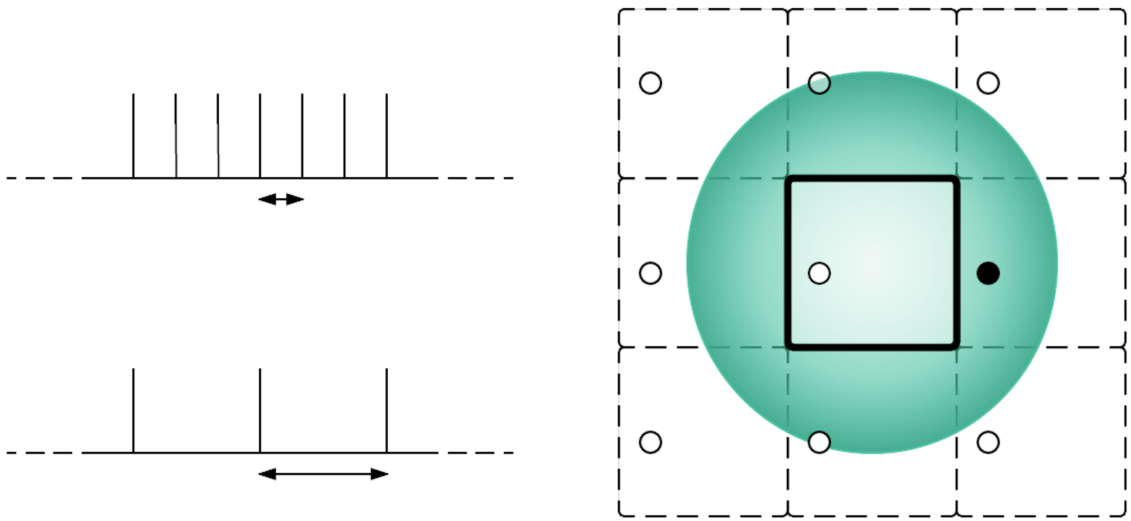
Interferometer data can be regularly spaced in time and frequency, but are not regularly spaced in u and v.

In order to use an FFT we need to GRID our data. This causes its own issues…

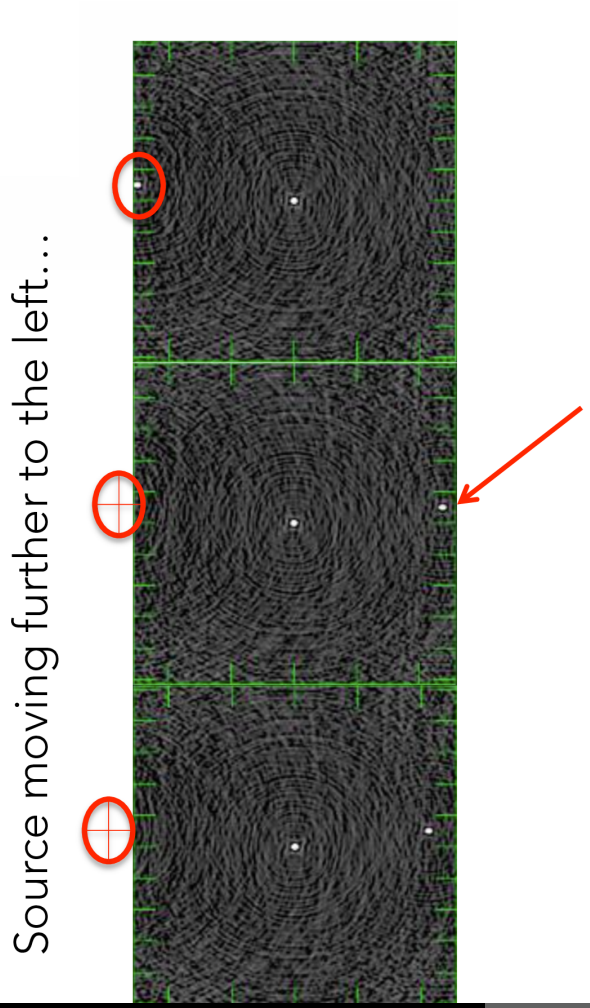
Gridding



Gridding



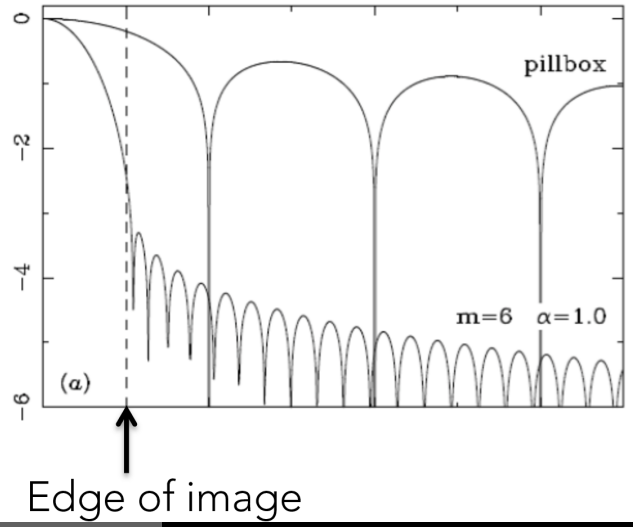
Aliasing



Aliasing can be prevented by using a suitable convolution function when we do the gridding.

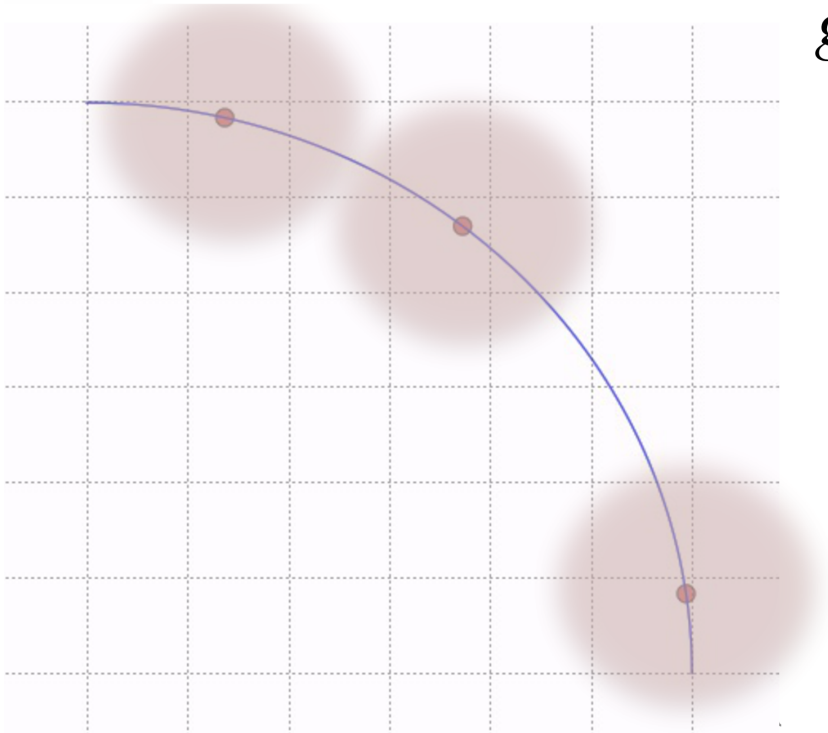
The most commonly used "anti-aliasing" kernel is the Prolate Spheroidal Wave Function (PSWF).

Casapy (toolkit):
`im.setoptions(gridfunction='sf')`



Anti-Aliasing

Convolution kernels cause each visibility to contribute to multiple pixels in our uv-grid.



$$g(x) \times f(x) \Leftrightarrow \tilde{g}(k) * \tilde{f}(k)$$

When we do the FFT our image will be multiplied by the FT of our convolution kernel.

$$I_{meas}(l,m) = \frac{F^{-1}[V_{grid}(u_k, v_k)]}{F^{-1}[C_{aa,grid}(u_k, v_k)]}$$

Anti-Aliasing

$$V_{grid}(u_k, v_k) = \left[[V(u,v) \cdot S(u,v)] * C_{aa}(u,v) \right] \cdot III(u_k, v_k)$$

The gridded visibility data on a grid with $\sqrt{k}$ x $\sqrt{k}$ pixels

The input visibility data, sampled at a number of times and frequencies.

The convolution/ gridding kernel function.

Sample onto regularly spaced grid using the *Shah* function.

Anti-Aliasing

$$V_{grid}(u_k, v_k) = \left[[V(u,v) \cdot S(u,v)] * C_{aa}(u,v) \right] \cdot III(u_k, v_k)$$

I have only considered the anti-aliasing kernel here; however, there are extra kernels that one can use to correct different effects.

Most imaging now uses w-projection (for correcting direction dependent effects); for w-projection we combine the w-kernel and the aa-kernel:

$$C(u,v) = C_{aa}(u,v) * C_w(u,v)$$

Visibility Weighting

We can change the angular response of our measurements by changing the sampling in time and frequency, but this requires additional observational data.

Another way of changing this response is to include an additional *weighting function* in the UV gridding.

We can adapt this function to bring out features on different scales in the same data set.

Visibility Weighting

$$V_{grid}(u_k, v_k) = \left[[V(u,v) \cdot S(u,v) \cdot W(u,v)] * C_{aa}(u,v) \right] \cdot III(u_k, v_k)$$

Fourier transform of sky intensity

Sampling function from baseline distribution

Visibility weighting function

Anti-aliasing kernel

Shah function (grid sampling)

Visibility Weighting

Natural **Robust** **Uniform**

$$W_k = 1 \qquad W_k = \frac{1}{[s \cdot \rho(u_k, v_k) + \sigma_k^2]} \qquad W_k = \frac{1}{\rho(u_k, v_k)}$$

Lower angular resolution

Better signal to noise

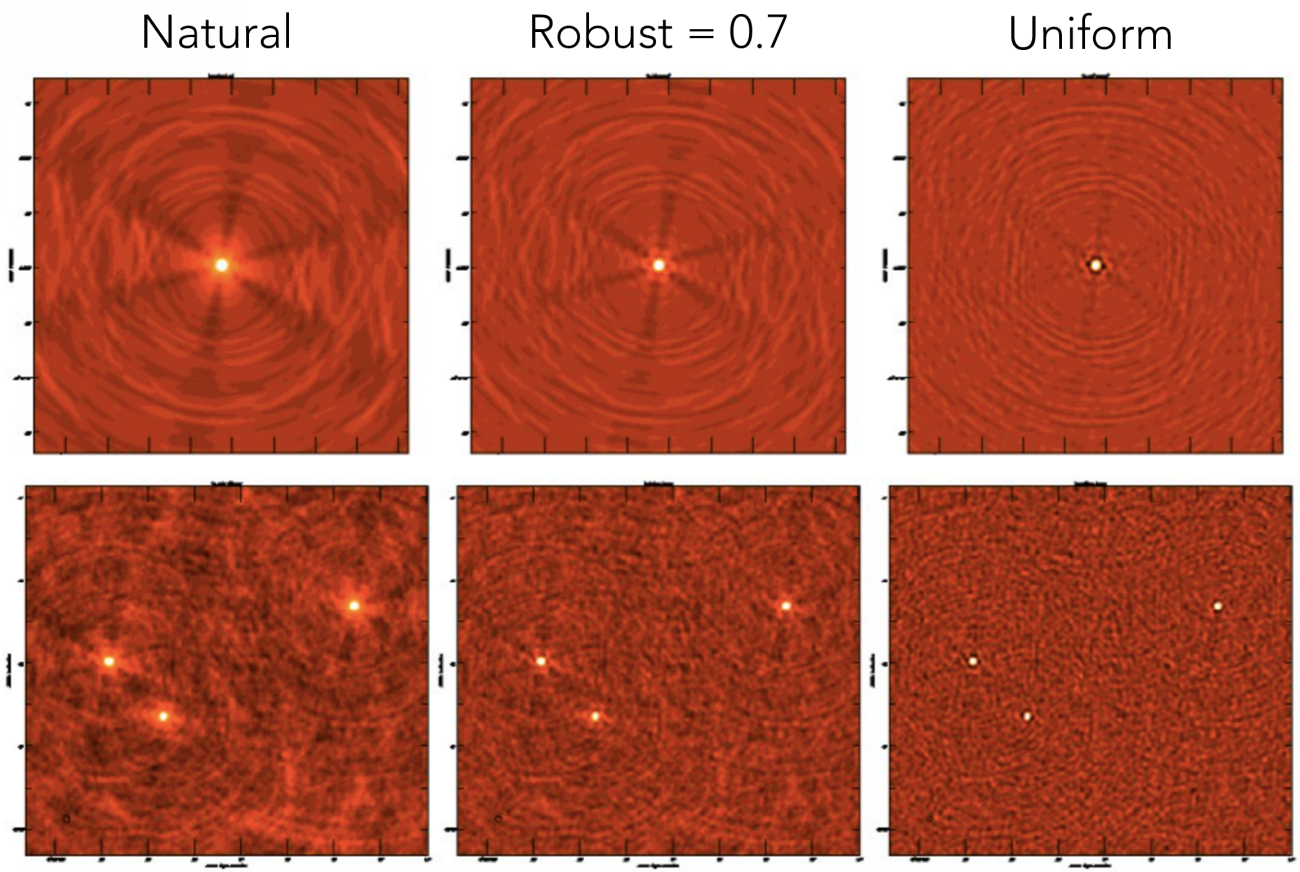
Poorer sidelobe structure

Higher angular resolution

Poorer signal to noise

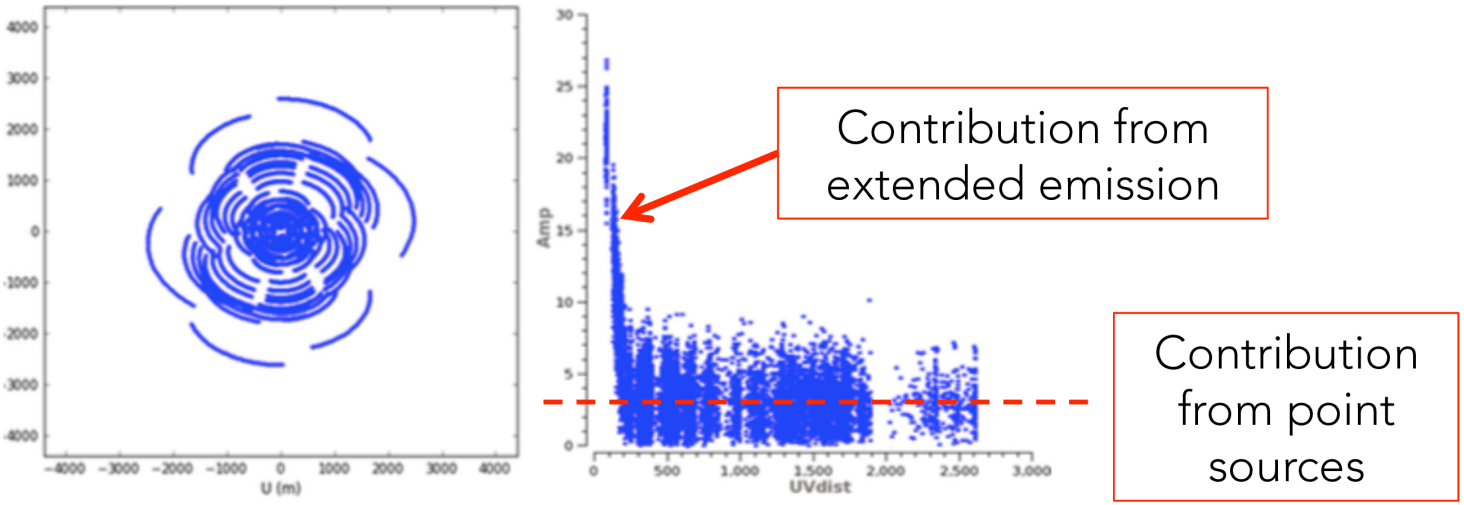
Better sidelobe structure

Visibility Weighting



Visibility Weighting

All of the examples so far show only point sources. To enhance the signal-to-noise of emission that is extended relative to the size of the PSF a weighting known as "uv-tapering" can be applied to the data.



Imaging

$$V_{grid}(u_k, v_k) = \left[[V(u,v) \cdot S(u,v) \cdot W(u,v)] * C_{aa}(u,v) \right] \cdot III(u_k, v_k)$$

To make an image we can now simply FFT our UV grid, but we must also correct for the gridding function that we have introduced and normalise the weights:

$$I_{\text{meas, dirty}}(l,m) = \frac{\text{FT}^{-1}[V_{\text{grid}}(u,v)]}{\left(\sum W_{\text{grid}}(u,v)\right) \text{FT}^{-1}[C_{\text{aa,grid}}(u,v)]}$$

The image that we have made is known as the DIRTY IMAGE, because we have not made any correction for the weighted sampling $S(u,v)W(u,v)$.

Because we are multiplying our continuous visibilities by $S(u,v)W(u,v)$ the DIRTY IMAGE shows us the convolution of their Fourier transforms.

$$V(u,v)\cdot\Big[S(u,v)\cdot W(u,v)\Big] \Leftrightarrow I(l,m) * b_{\mathrm{PSF}}(l,m)$$

Where the point spread function, or synthesized beam, or dirty beam, is defined as

$$b_{\mathrm{PSF}}(l,m) = FT^{-1}\Big[S(u,v)\cdot W(u,v)\Big]$$

It would be nice if we could just divide out this multiplication directly in Fourier space, but we can't because it has zero-valued components.

Deconvolution

We can correct (partially) for the effect of the synthesized beam by deconvolving it using an iterative algorithm.

The most widely used deconvolution method in radio interferometry is the CLEAN algorithm.

There are multiple variants of CLEAN: Hogbom, Clark, Cotton-Schwab, Multi-scale, ASP…

The simplest variant of CLEAN is called Hogbom CLEAN.

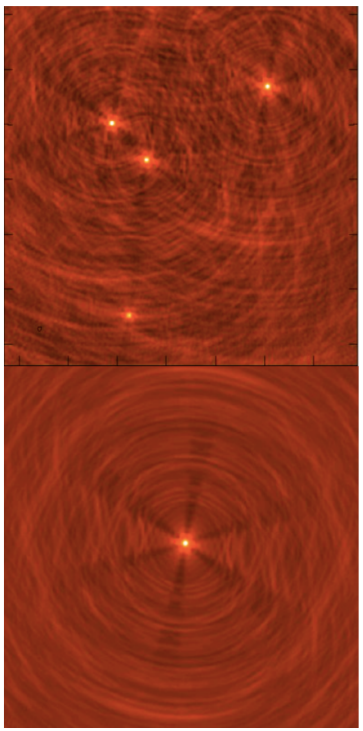
Deconvolution

This algorithm works entirely on the dirty image. It iteratively subtracts the PSF from the image until a stopping criterion is met.

Assumptions:

- The sky intensity can be well represented by a sum of delta functions.

$$I_{true} = \sum_{j=1}^{N_{src}} a_j \delta(l - l_j, m - m_j)$$



Deconvolution

This algorithm works entirely on the dirty image. It iteratively subtracts the PSF from the image until a stopping criterion is met.

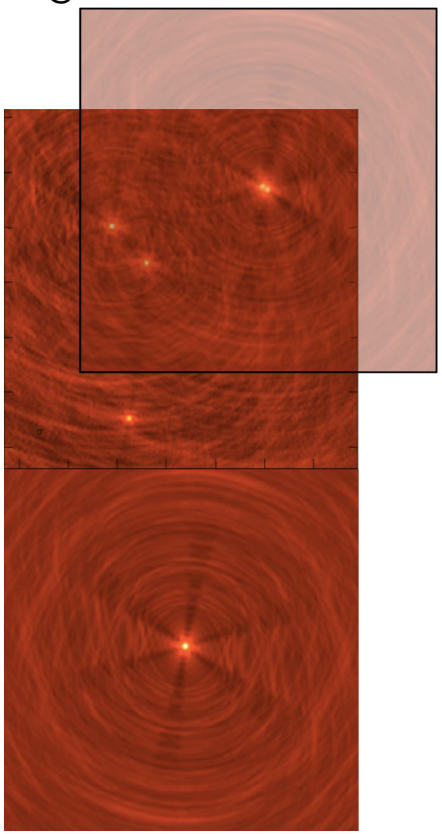
Assumptions:

- The sky intensity can be well represented by a sum of delta functions.

Preparation:

- Make your dirty image.
- Make your dirty beam.

  Note: *Your dirty beam image must be twice as large as your image.*

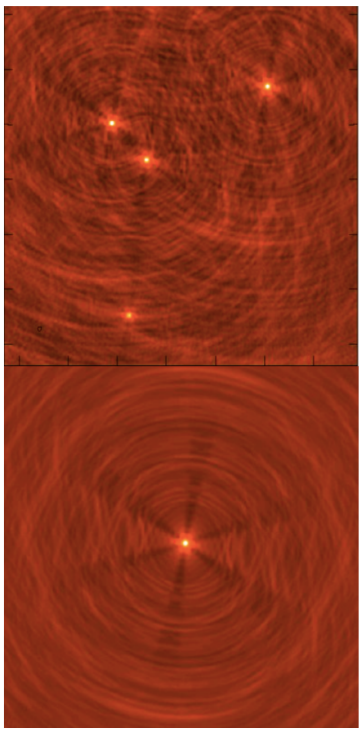


Deconvolution

Method:

Minor Cycle

1. Find the peak in the dirty image.
2. Subtract f x PSF at that location.
   f is called the "loop gain".
3. Store the position and peak value at that point.
4. Repeat 1-3 until the stopping criterion is met.

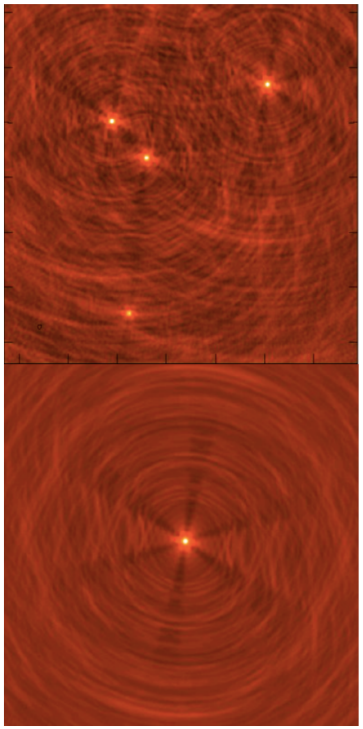


Deconvolution

This algorithm works entirely on the dirty image. It iteratively subtracts the PSF from the image until a stopping criterion is met.

Stopping Criteria:

- Number of iterations (repeats).
- Negative peak identified.
- Smallest peak that can be identified is below some threshold.
    - Hard threshold.
    - Noise based threshold.

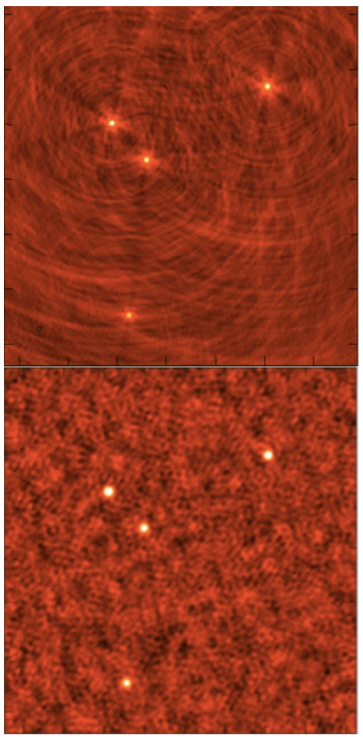


Deconvolution

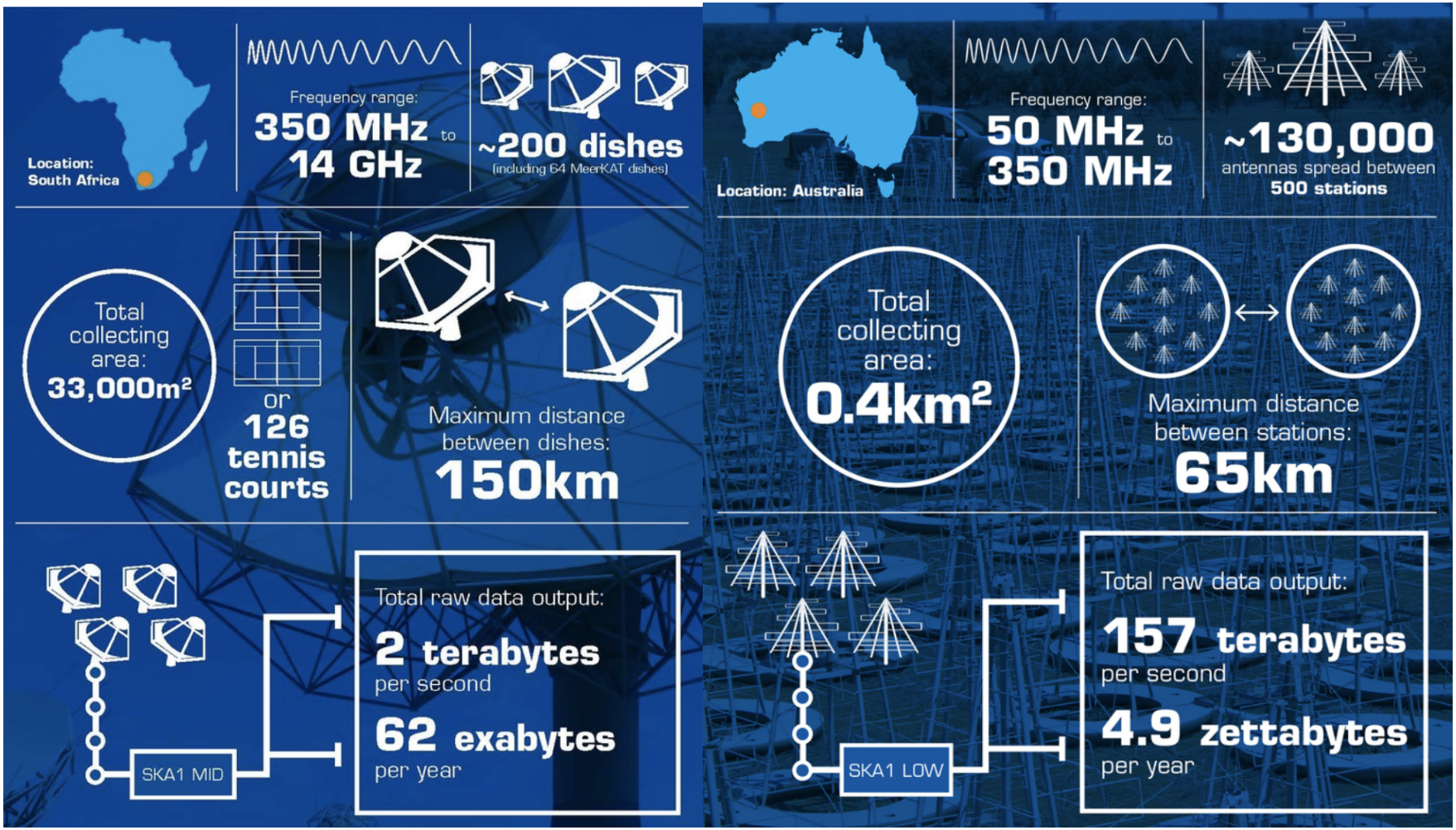
This algorithm works entirely on the dirty image. It iteratively subtracts the PSF from the image until a stopping criterion is met.

Outputs:

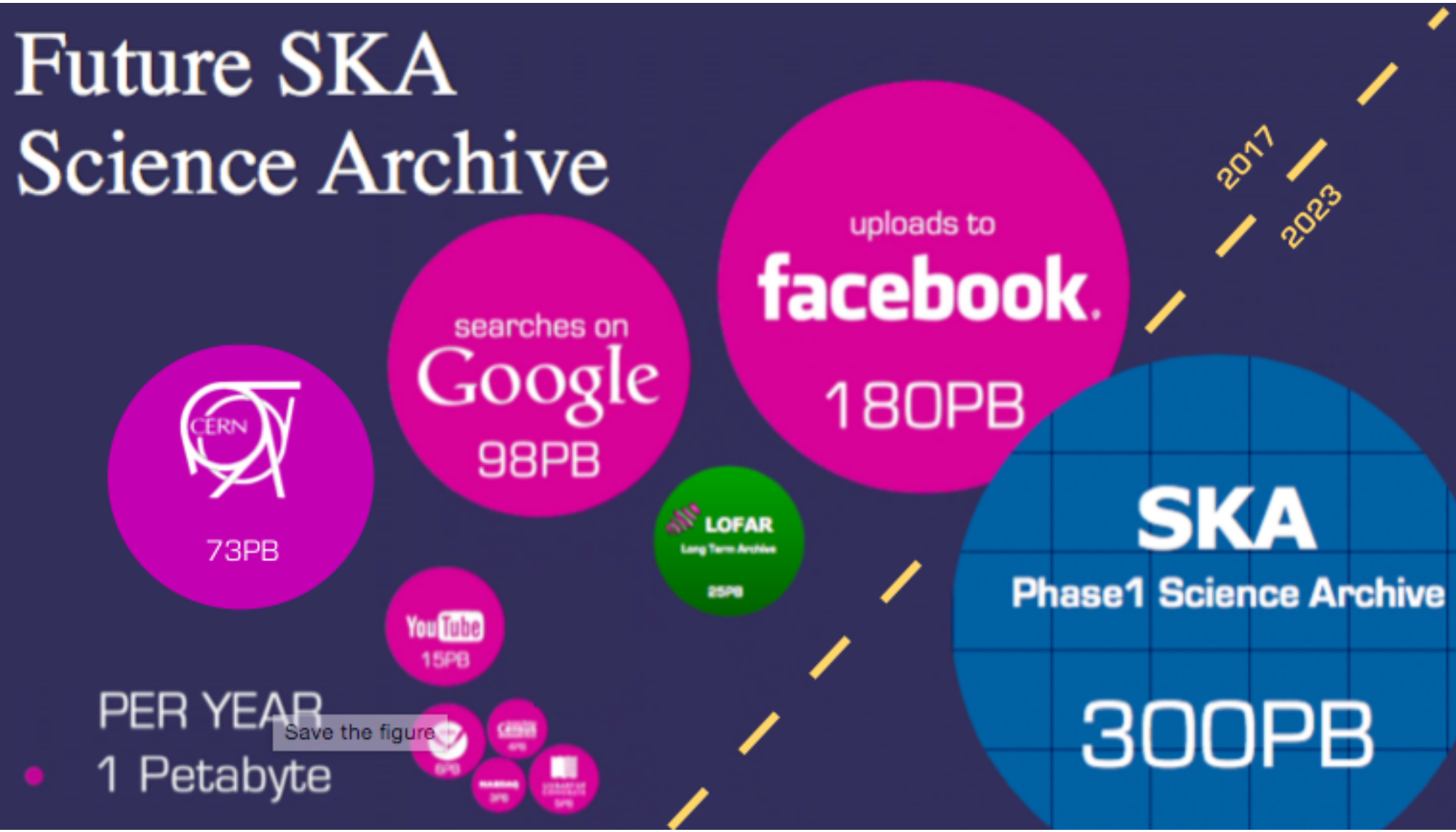
- When you stop iterating (for whatever reason) you call the final subtracted image your RESIDUAL IMAGE.
- Add a Gaussian with the width of the main peak of the dirty beam ("CLEAN BEAM") to the residual image for each position and peak value in your list of subtracted components (known as "clean components") to make the CLEAN or RESTORED IMAGE.



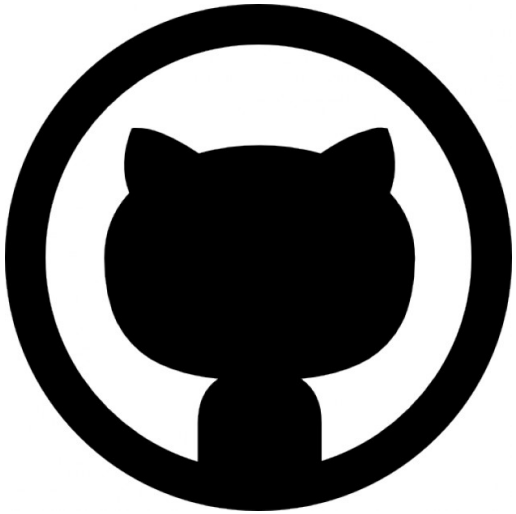
SKA Data Volumes



SKA Data Volumes



Tutorial Material



`https://github.com/as595/NIThePtree/master/TUTORIALS`

`SimulateInterferometer.ipynb`