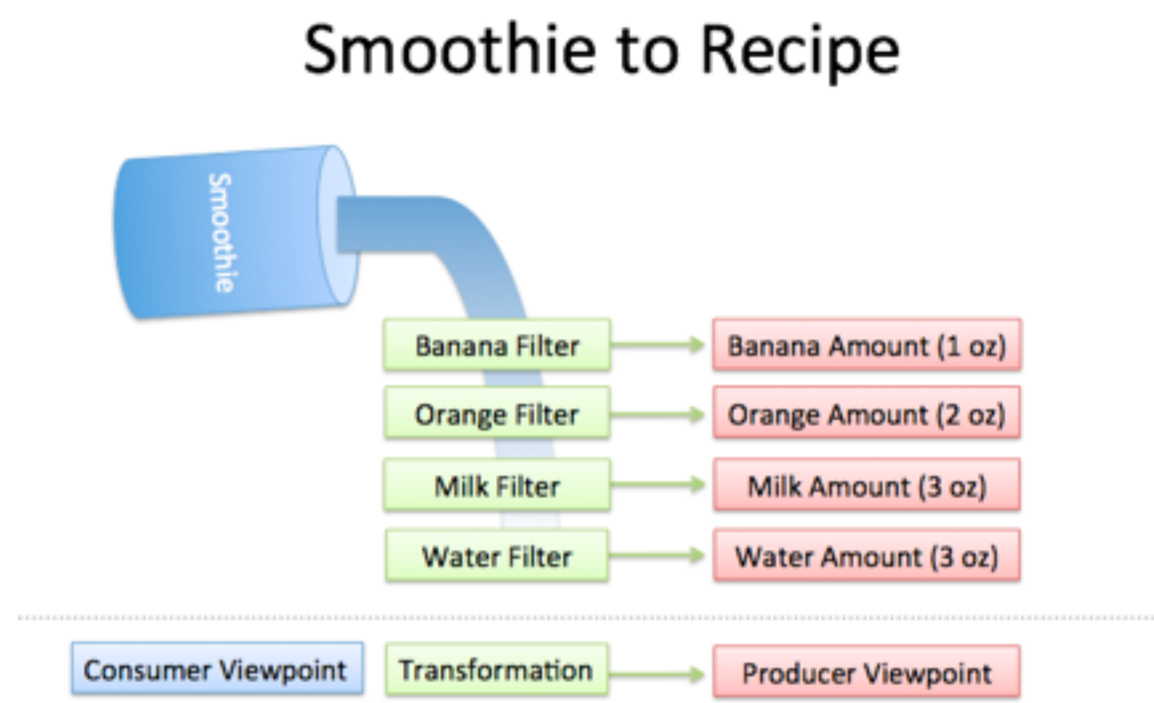
Fun with Fourier Transforms

Introduction



- **What does the Fourier Transform do?**
  - Given a smoothie, it finds the recipe
- **How?**
  - Run the smoothie through filters to extract the ingredients
- **Why?**
  - Recipes are easier to analyse, compare and modify
- **How do we get the smoothie back?**
  - We blend the ingredients

The Fourier Transform takes a time-based pattern, measures every possible cycle, and returns the overall "cycle recipe" (the strength, offset, & rotation speed for every cycle that was found).

Introduction

Some useful properties of Fourier transforms in 1-D

$$
\begin{aligned}
F(\nu) &= \int_{-\infty}^{\infty} f(t)\exp(-2\pi i\nu)dt \\
f(t) &= \int_{-\infty}^{\infty} F(s)\exp(2\pi i\nu)d\nu
\end{aligned}
$$

Inversion

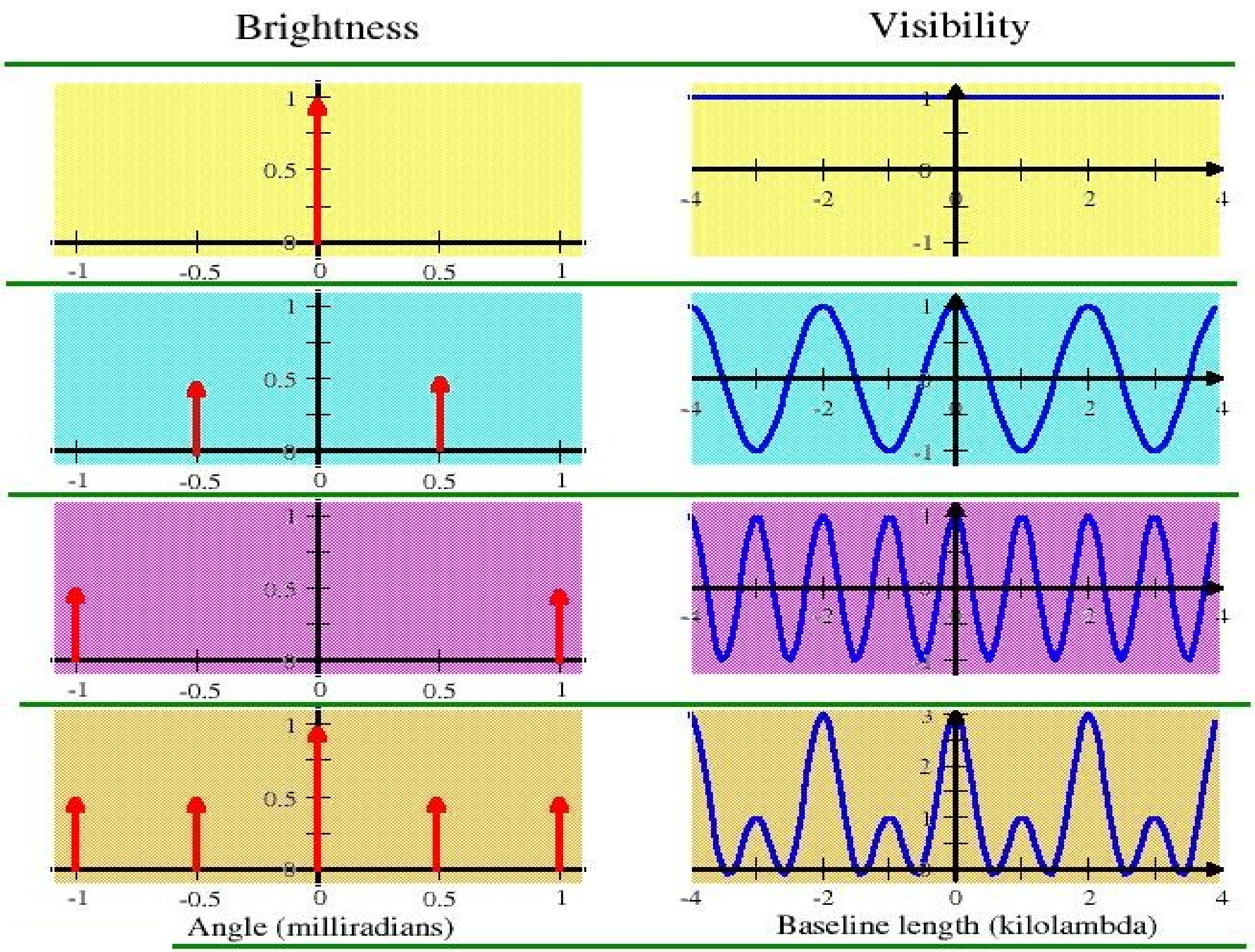
$$
\begin{aligned}
h(t) &= \int_{-\infty}^{\infty} f(t')g(t-t')dt' \\
H(\nu) &= F(\nu)G(\nu)
\end{aligned}
$$

Convolution

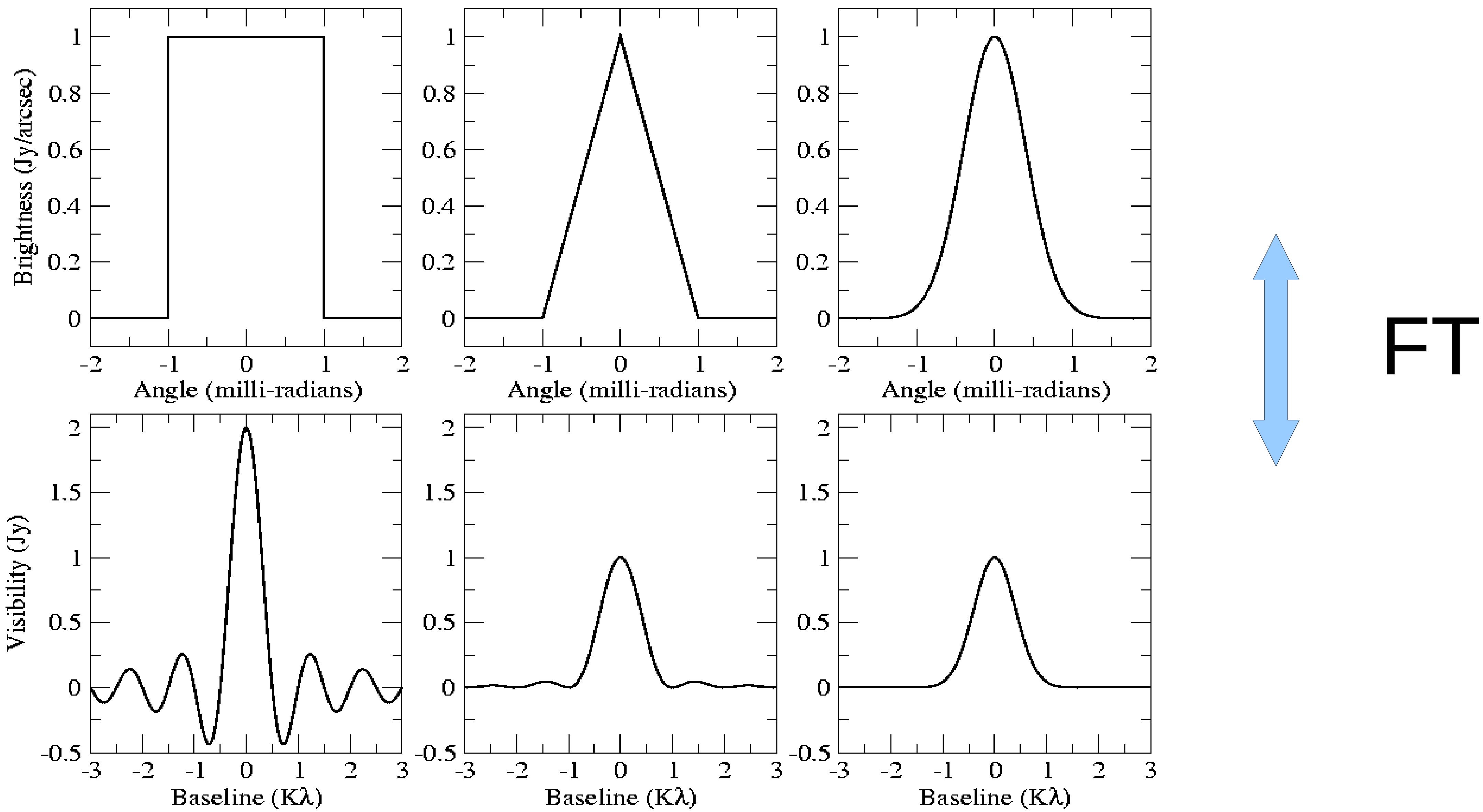
1-D examples

$$F(\nu) = \int_{-\infty}^{\infty} f(t)\exp(-2\pi i\nu)dt$$

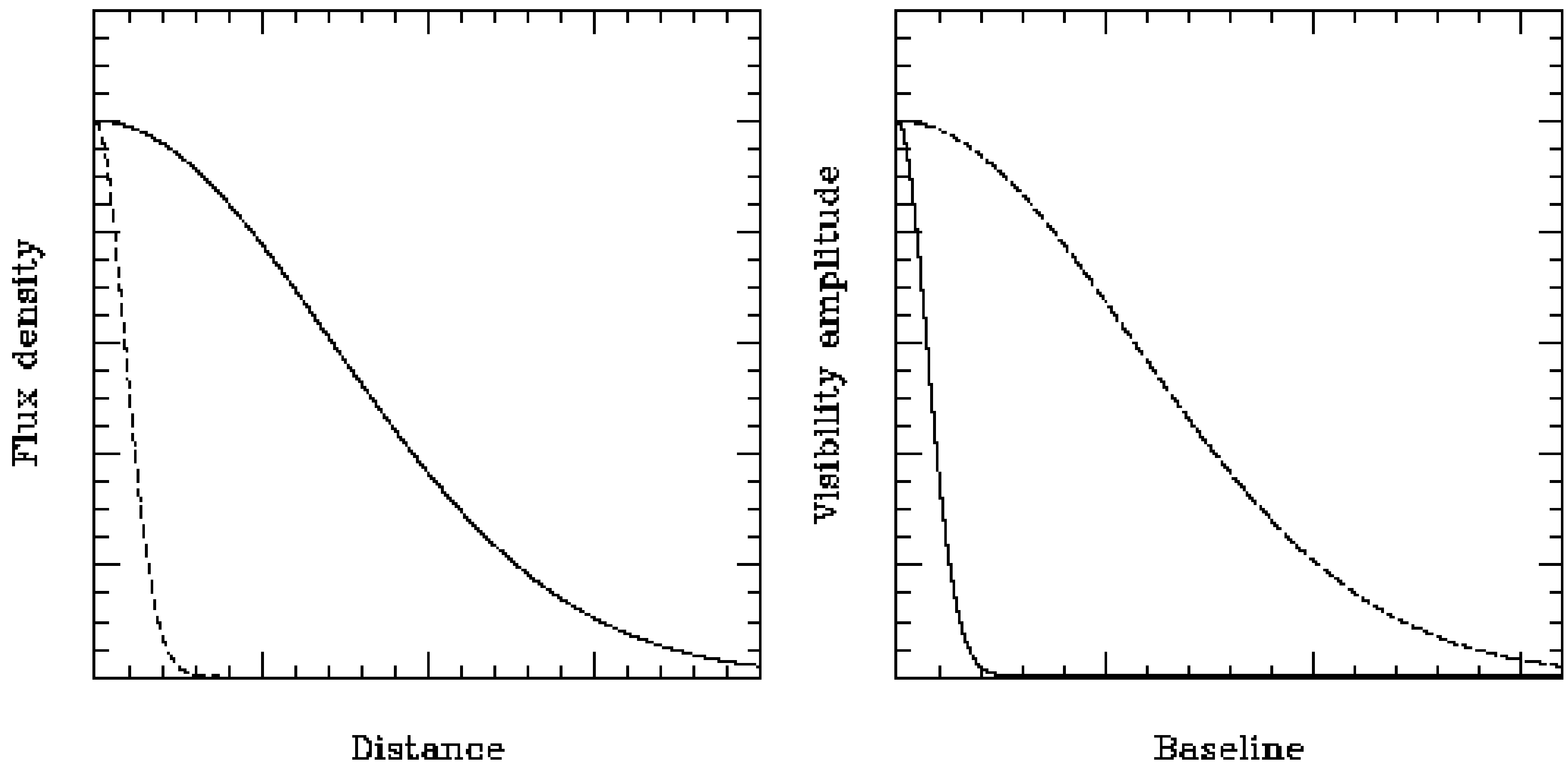
$$f(t) = \int_{-\infty}^{\infty} F(s)\exp(2\pi i\nu)d\nu$$



1-D examples



Note sharp edges in the image give ripples in the visibilities



- The Fourier transform of a Gaussian function is another Gaussian.
- FWHM on the sky is inversely proportional to FWHM in spatial frequency: fat objects have thin Fourier transforms and vice versa

2-D Examples

Definition

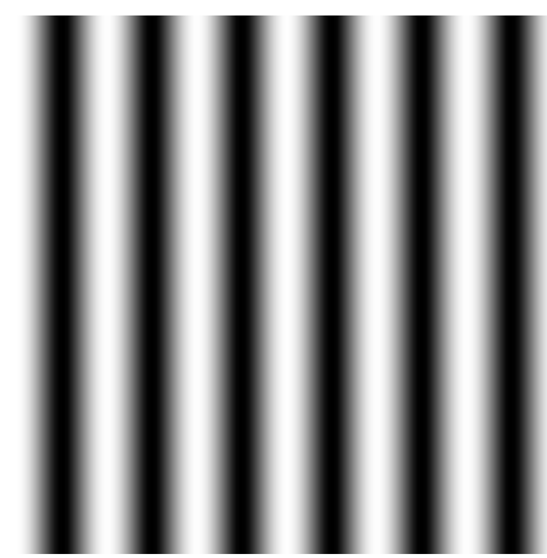
$$F(u,v) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x,y)e^{-j2\pi(ux+vy)}\,dx\,dy,$$
$$f(x,y) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} F(u,v)e^{j2\pi(ux+vy)}\,du\,dv$$

Where *u* and *v* are spatial frequencies

- F(u,v) is complex it has an amplitude and phase
- |F(u,v)| is the **magnitude** spectrum and tells us "how much" of a certain frequency component is present
- *arctan*($F_I(u,v)$ / $F_R(u,v)$) is the phase angle spectrum and it tells us "where" the frequency component is in the image

(Very) Basic Principles

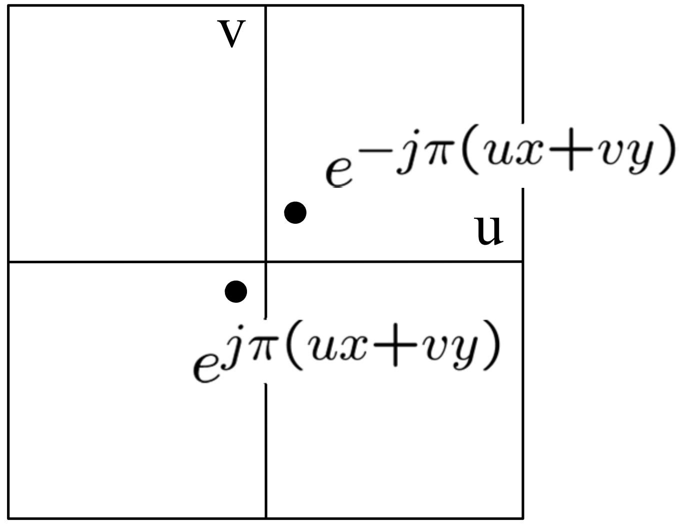
- The FT states that images can be expressed as the sum of a series of sinusoids, encoding the **spatial frequency**, **magnitude** and **phase**.

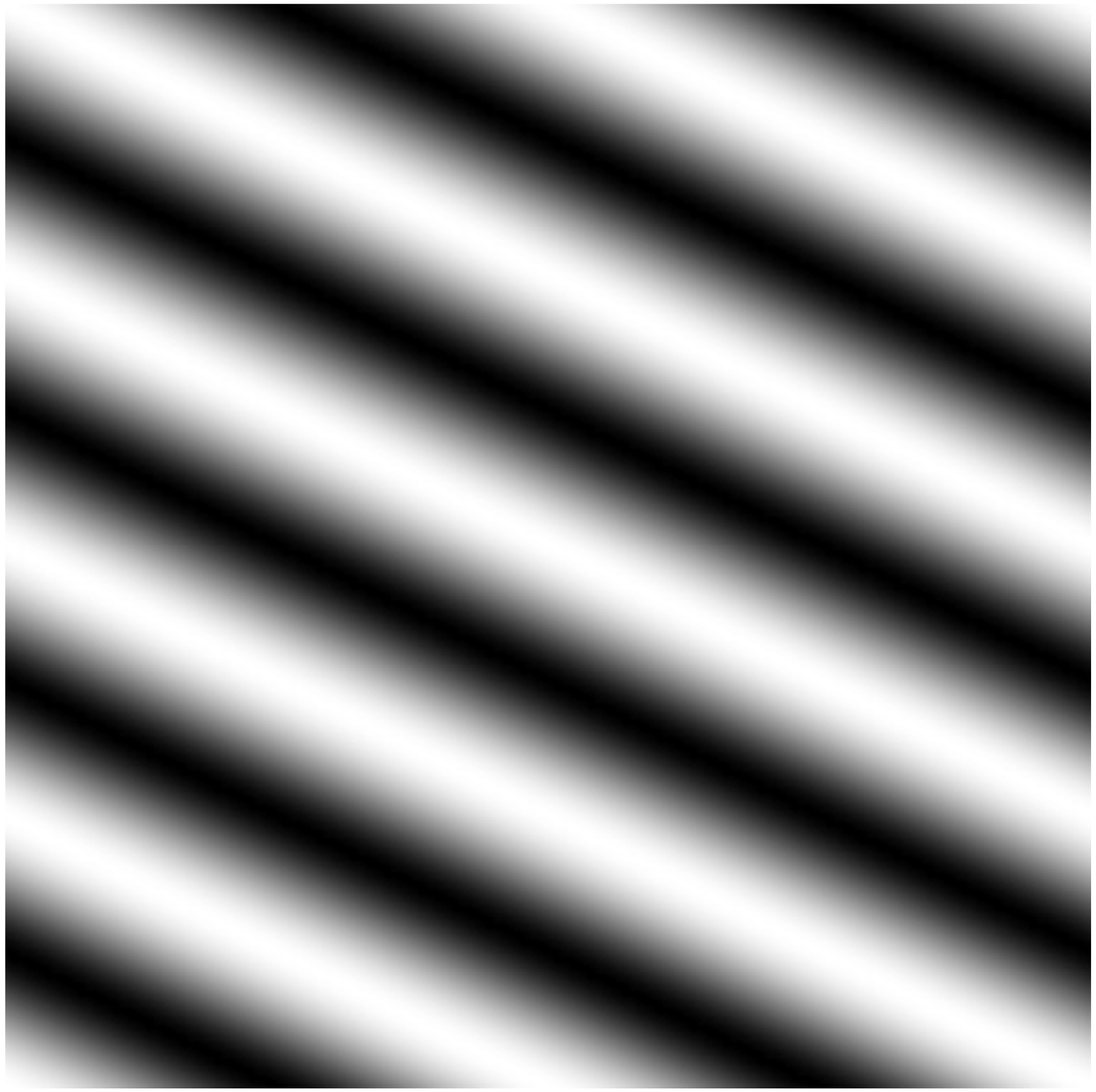


higher spatial frequency

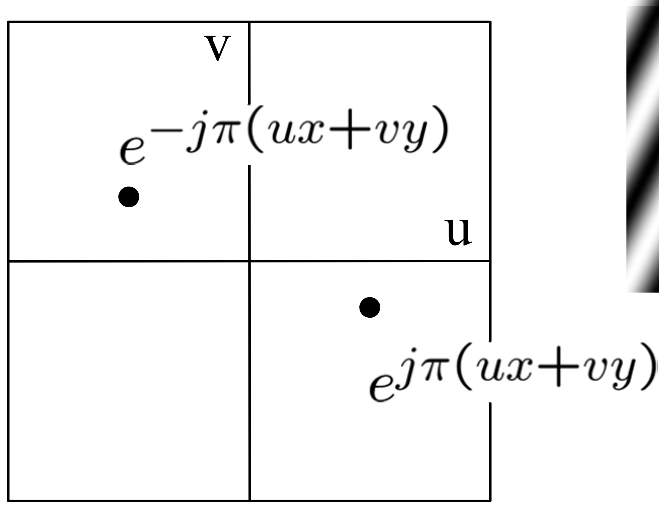
To get some sense of what basis elements look like, we plot a basis element --- or rather, its real part ---

as a function of x,y for some fixed u, v. We get a function that is constant when (ux+vy) is constant. The magnitude of the vector (u, v) gives a frequency, and its direction gives an orientation. The function is a sinusoid with this frequency along the direction, and constant perpendicular to the direction.

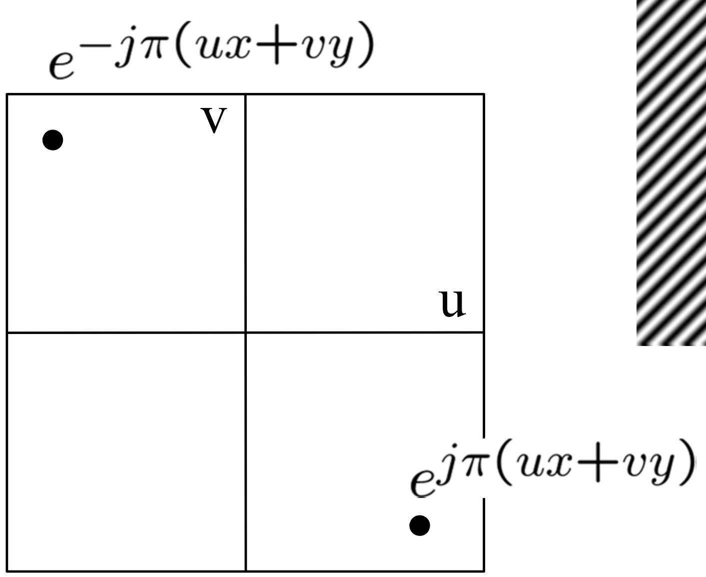


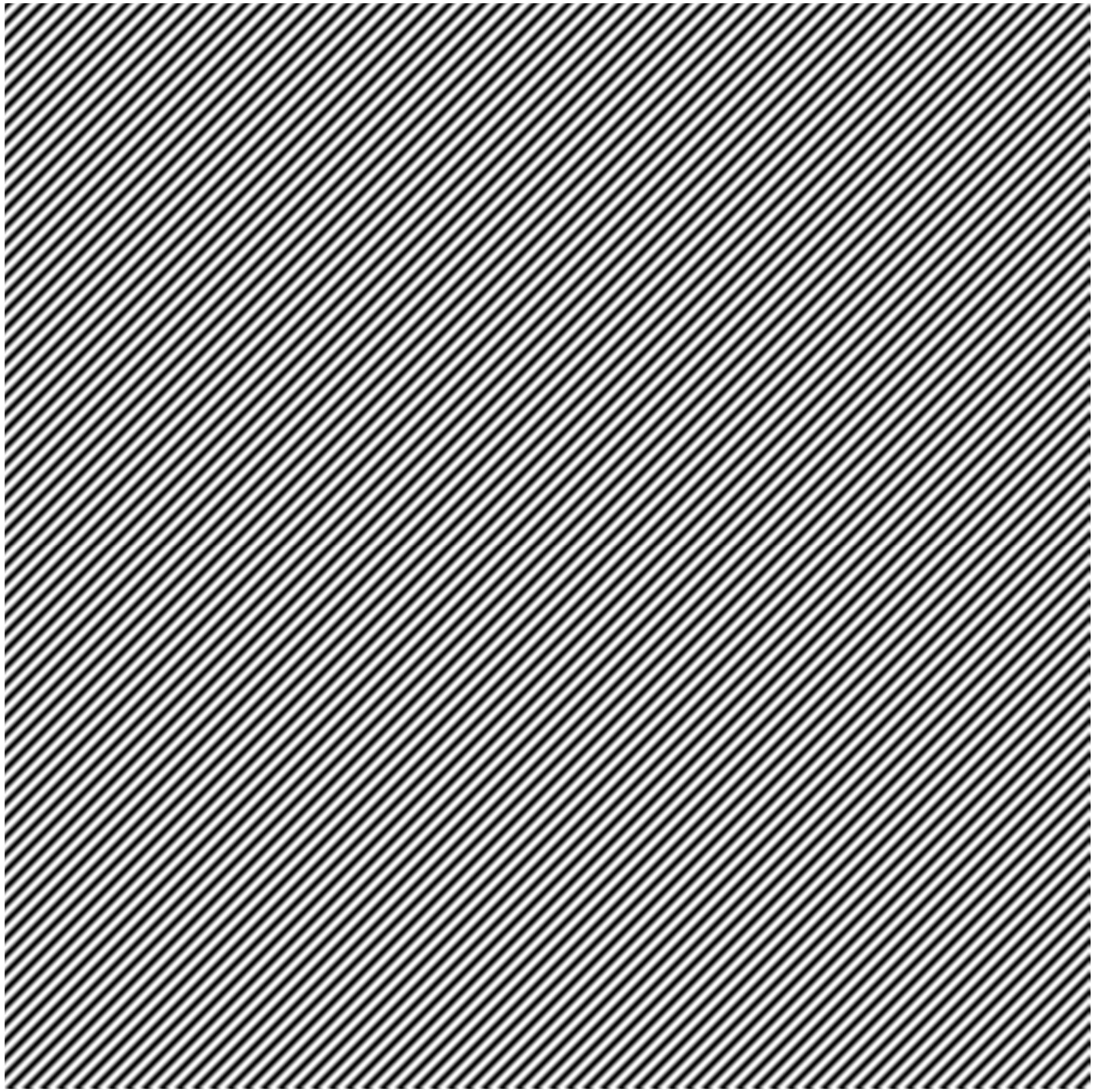


If we increase the distance between the two elements the fringe separation increases



Increasing the distance even further

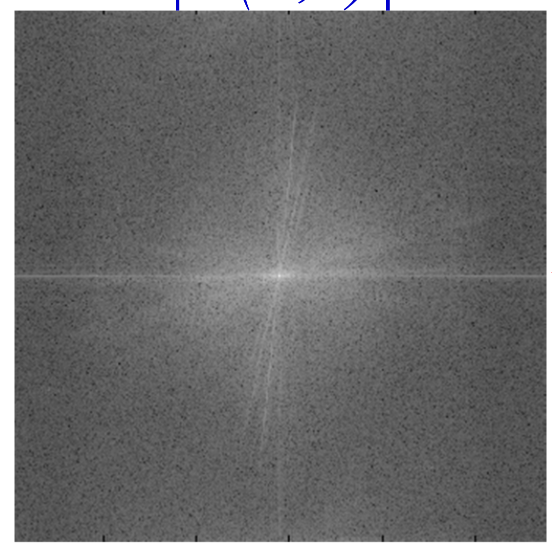




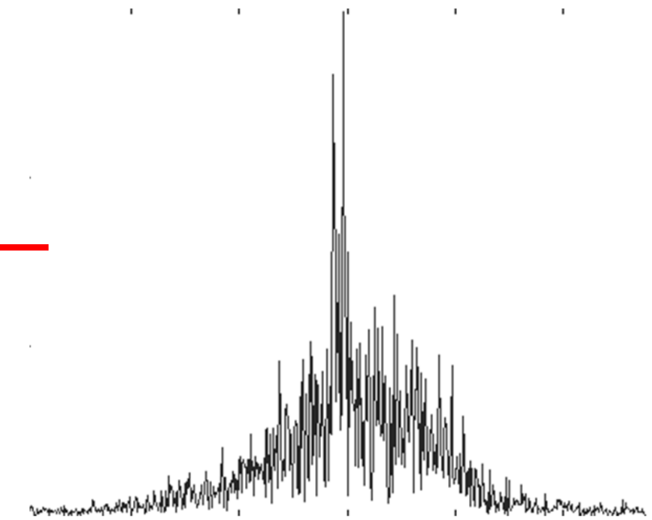
Magnitude vs Phase

f(x,y)

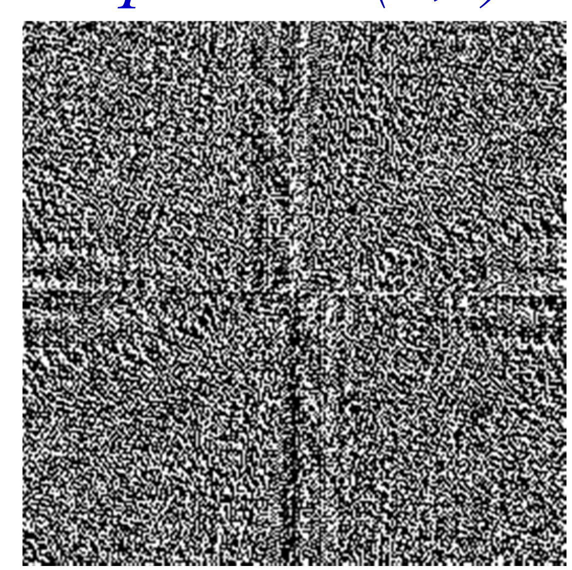
|F(u,v)|



cross-section



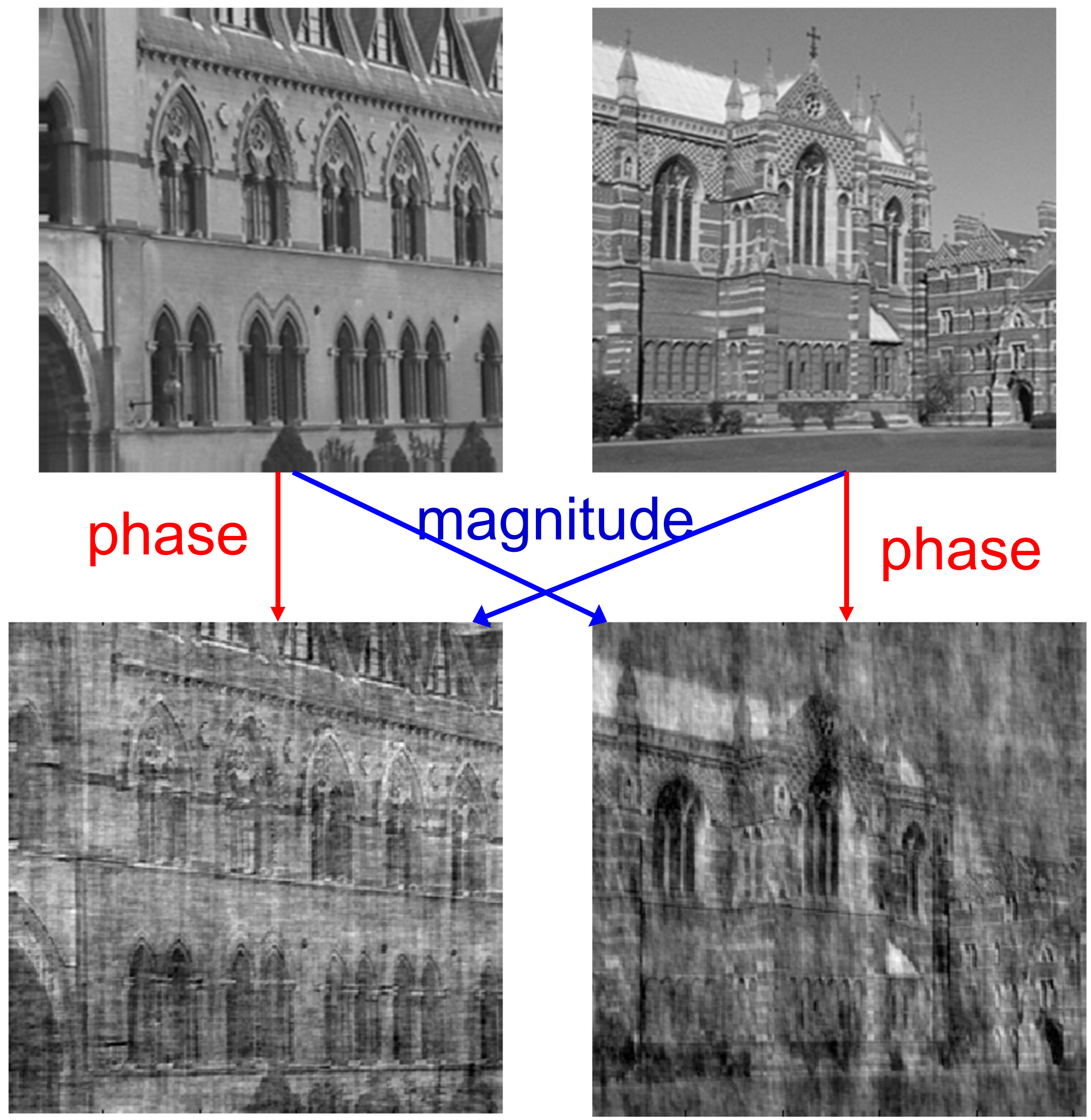
phase F(u,v)



- |F(*u*,*v*)| generally decreases with higher spatial frequency
- Phase appears less informative

The importance of phase

- Here we swap the phases of the two top images
- When we try to recover the images with the incorrect phase we see they are corrupt



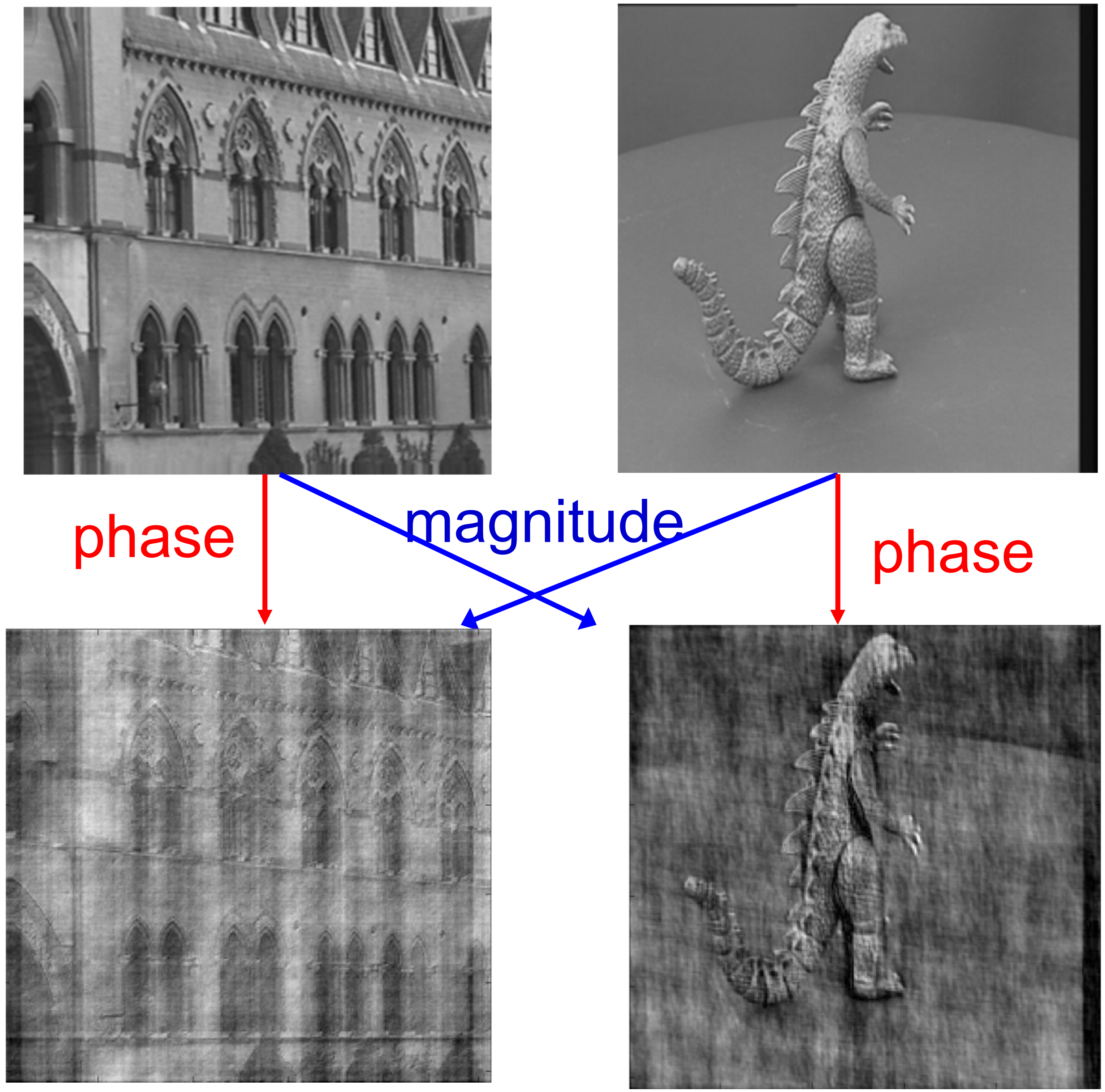
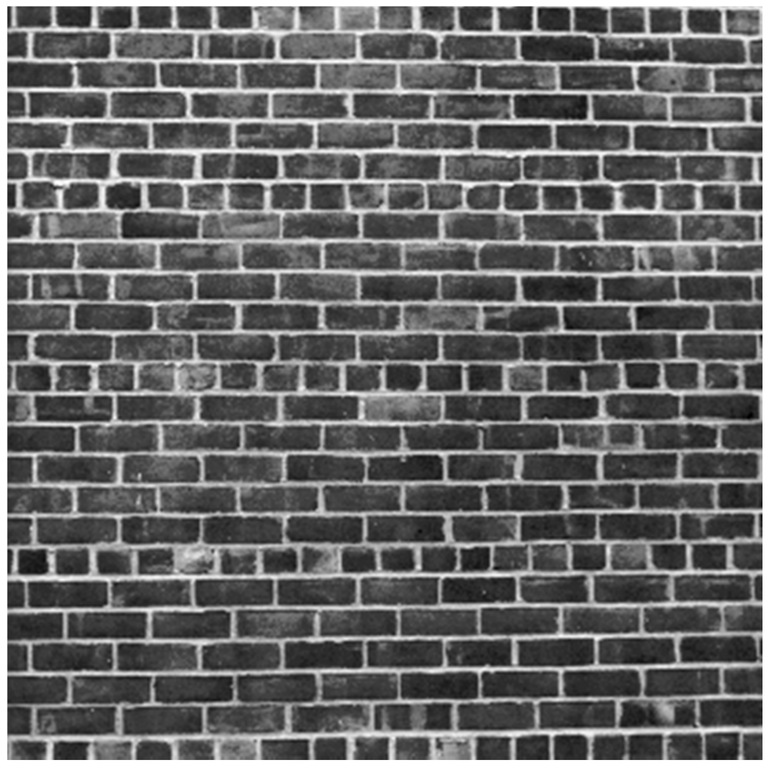
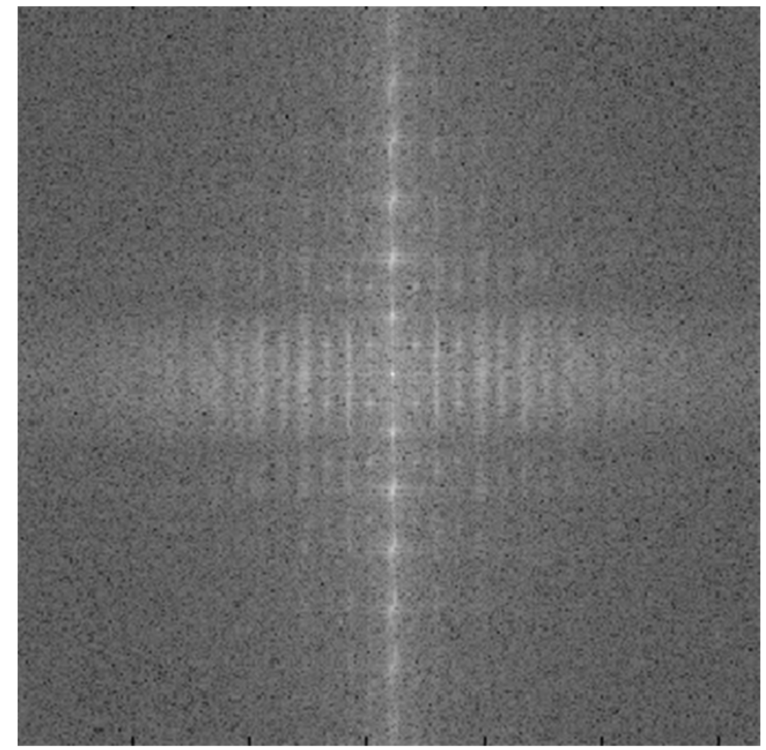

phase
magnitude
phase

Image with periodic structure



$f(x,y)$



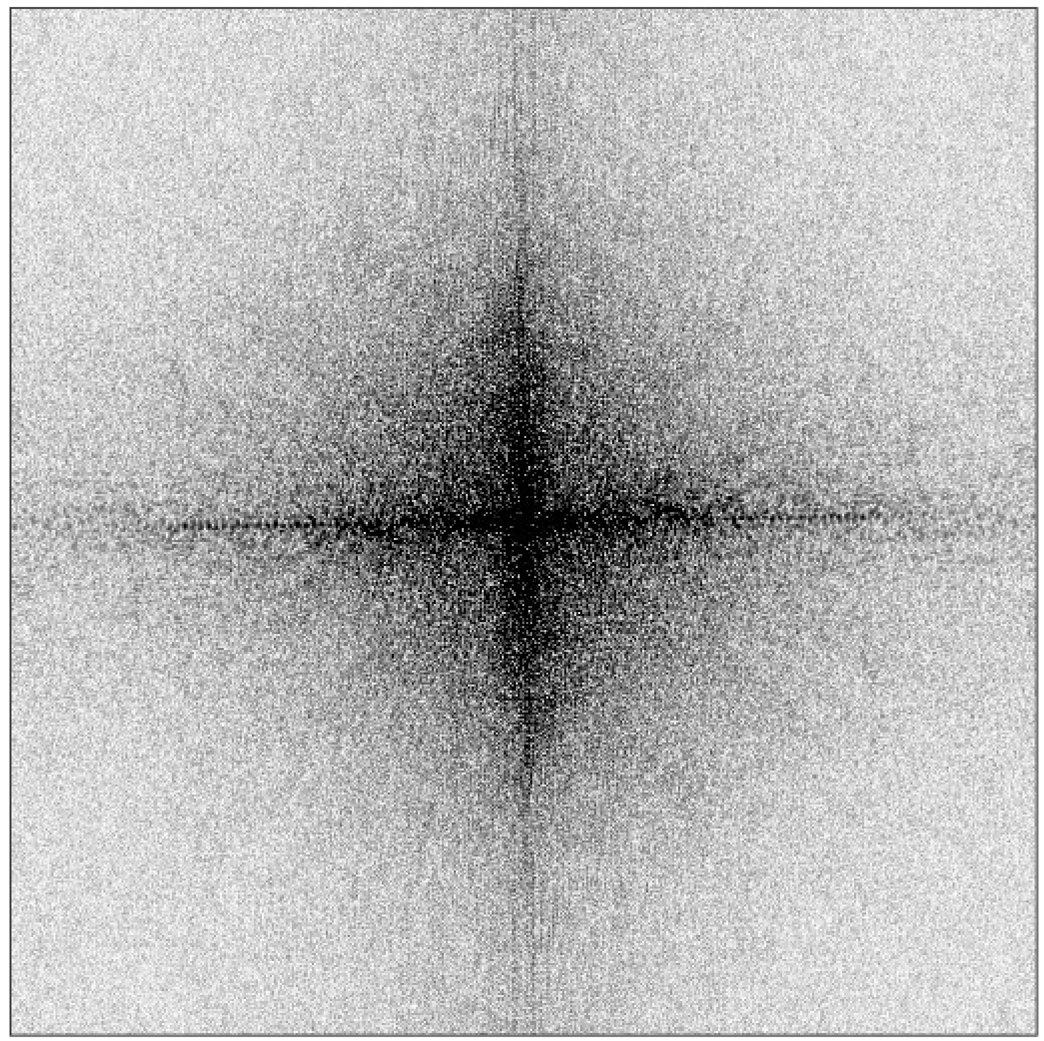
$|F(u,v)|$

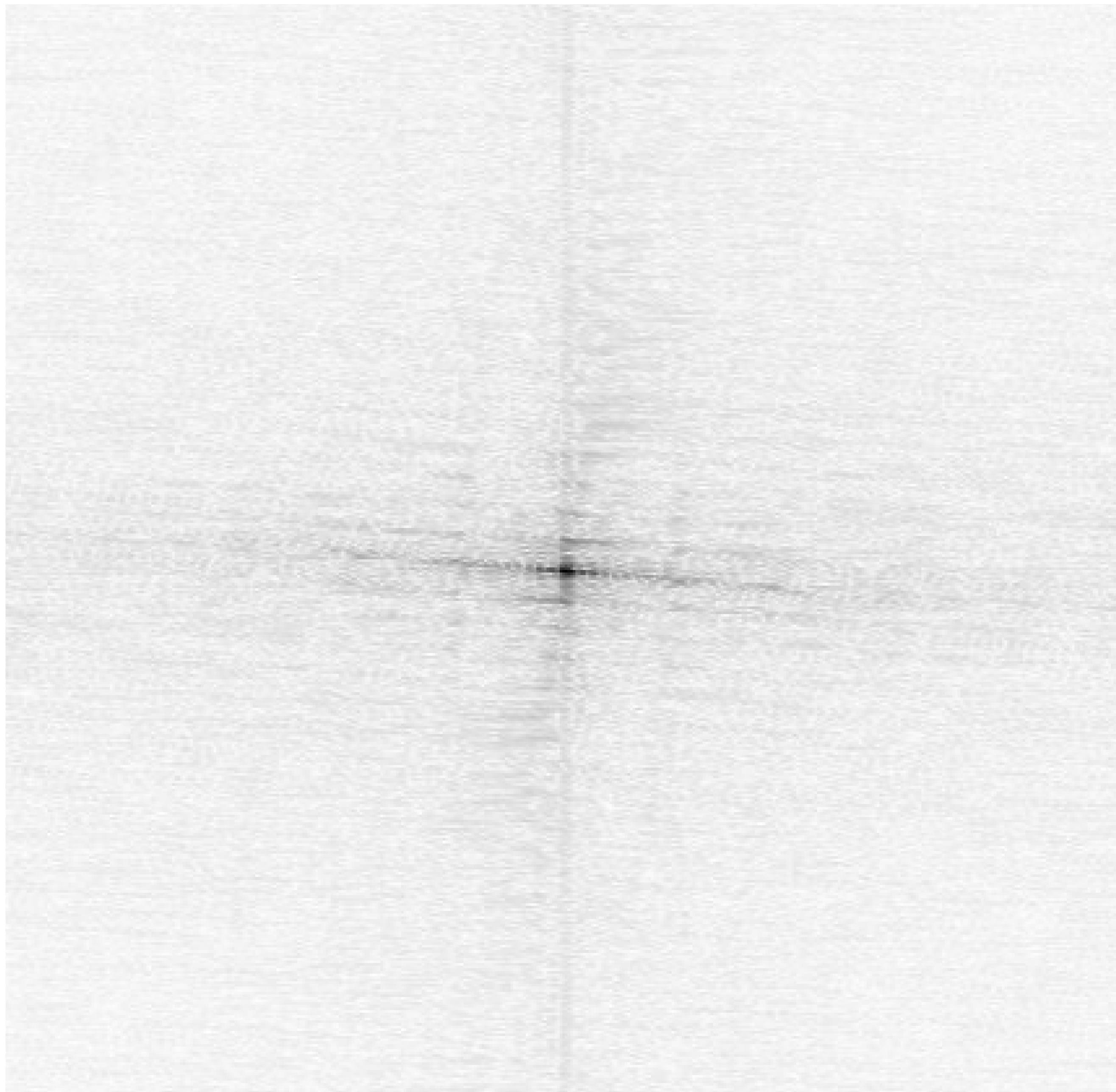
FT has peaks at spatial frequencies of repeated texture

What is this?

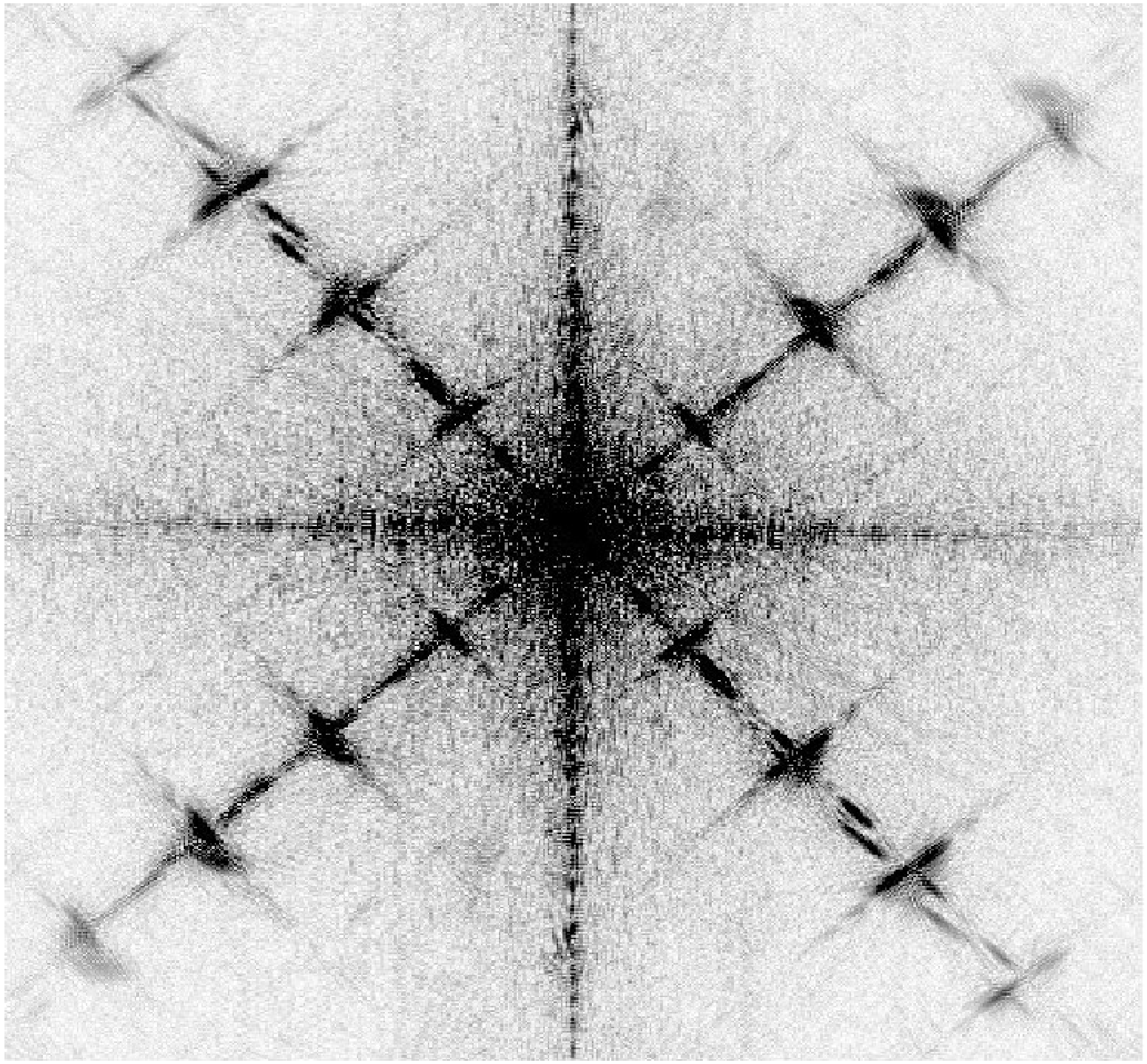
This is the amplitude of the FT of an image of a well-known local object

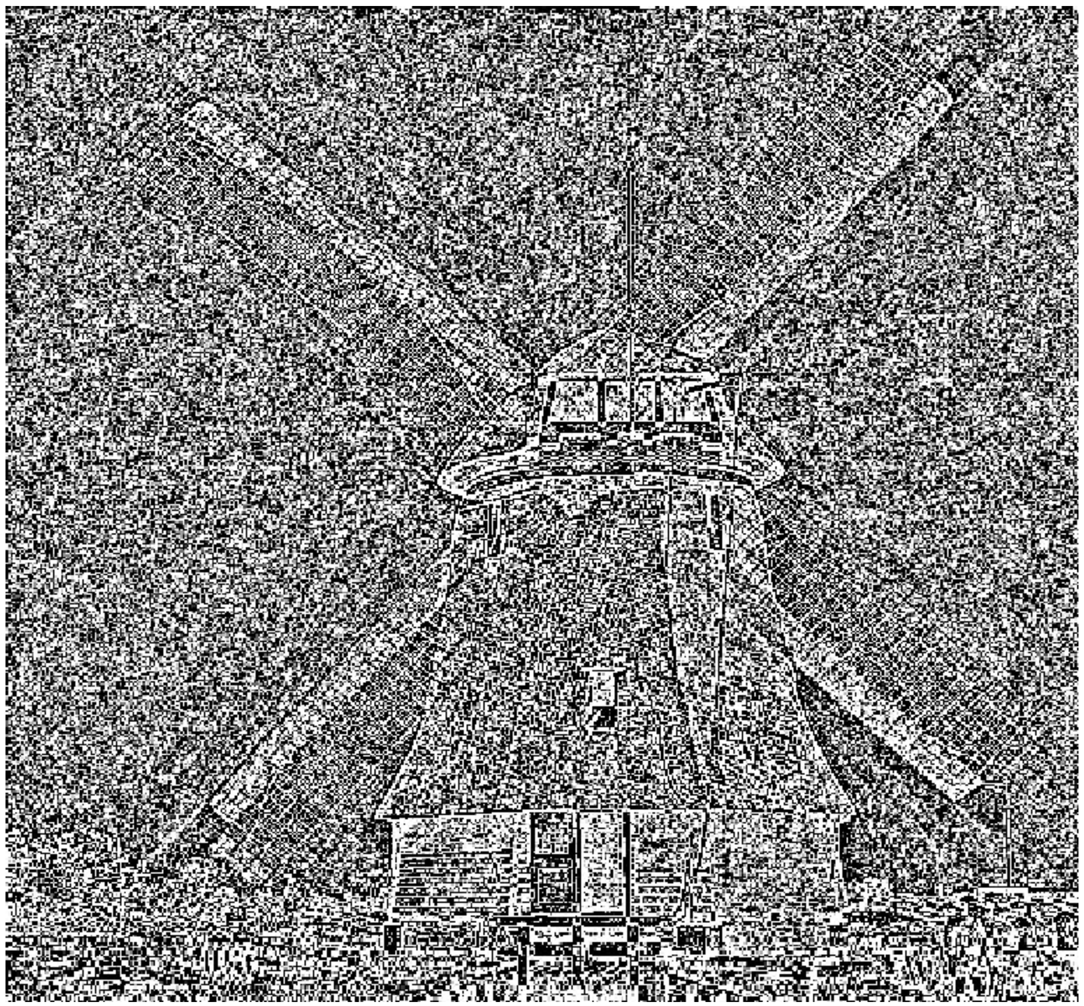
Can you say anything about it's size, shape, orientation?



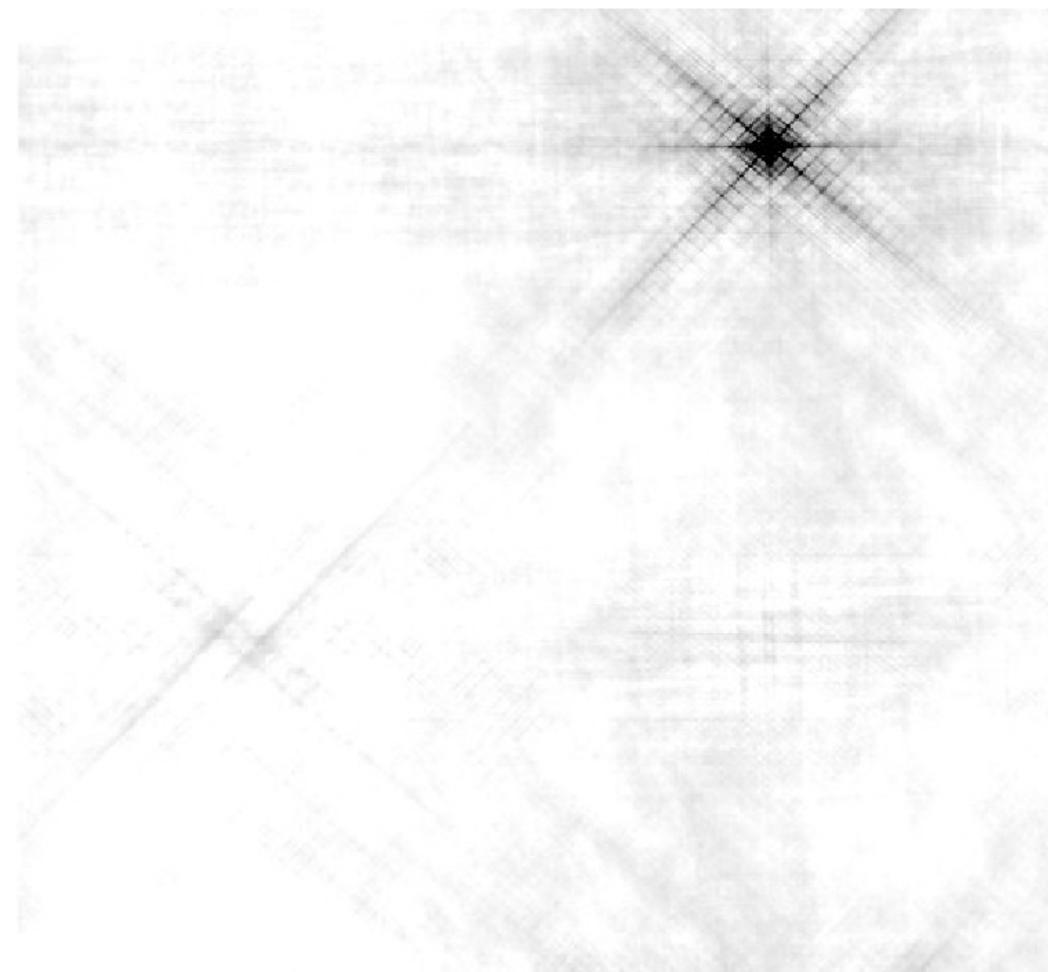


Can you say anything about it’s fine scale structure, size, shape and orientation?





Unit amplitude + correct phase

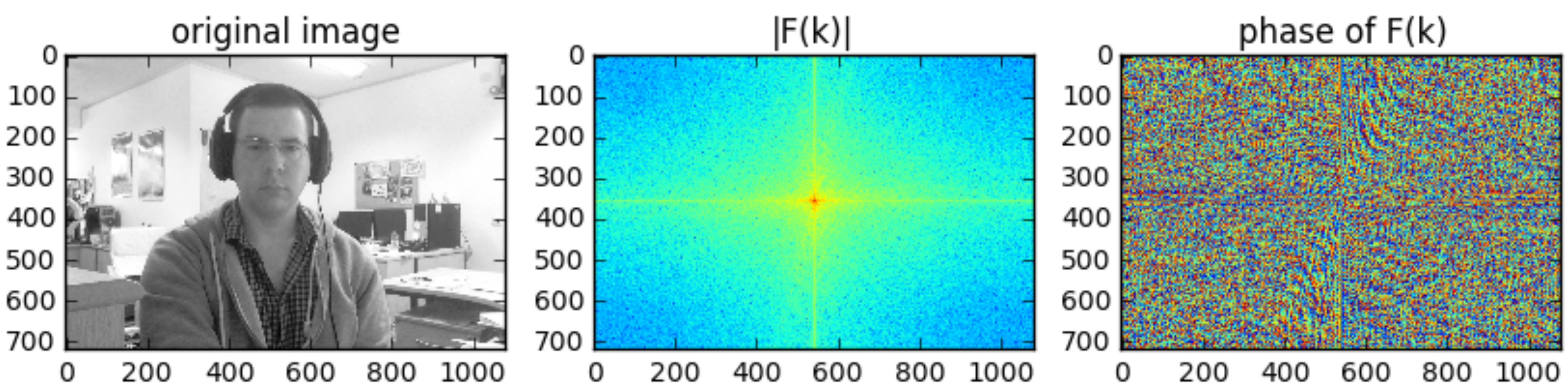


Zero Phase + correct amplitude

Your own Fourier Transform

- **FunWithFourier.py** uses the **numpy** module in python is able to perform Fourier transforms

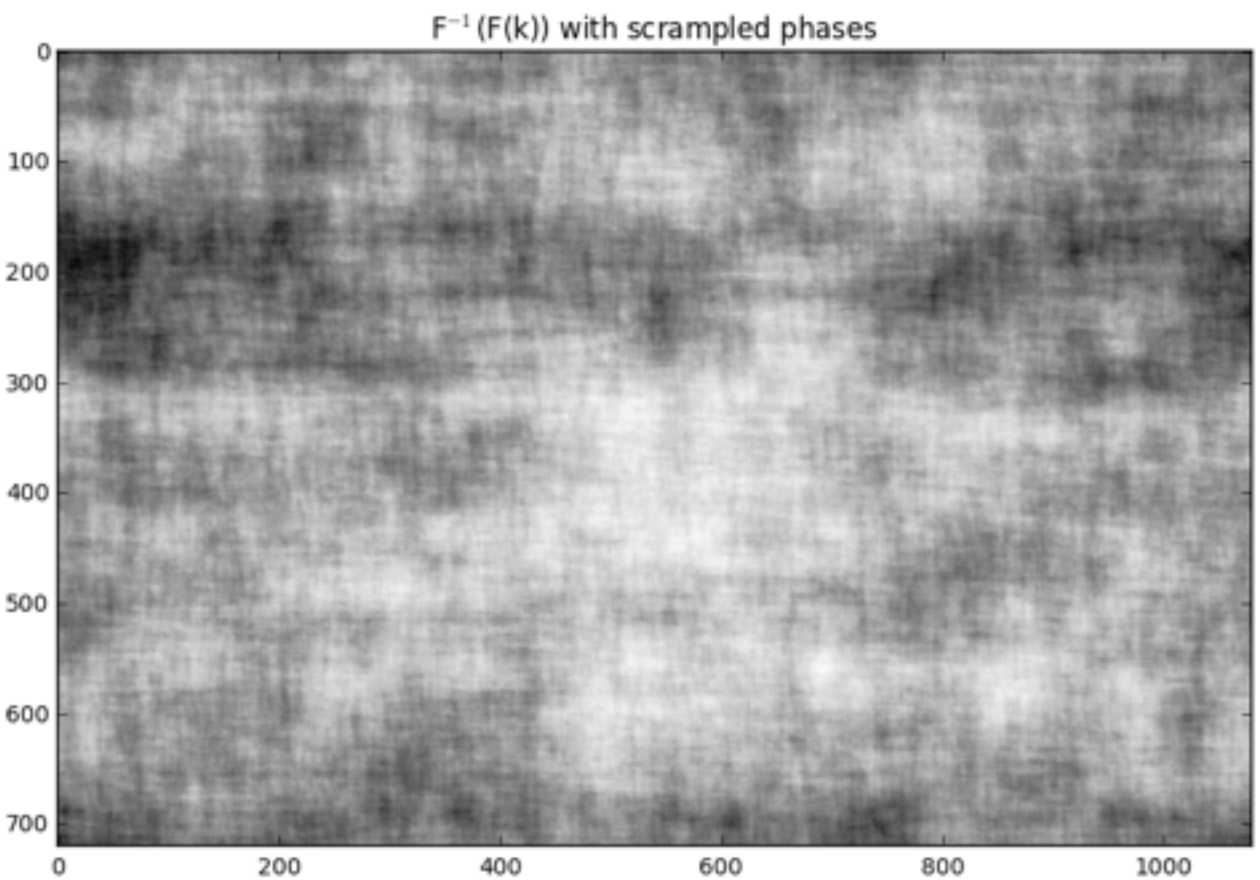
- The code produces an amplitude and phase plot

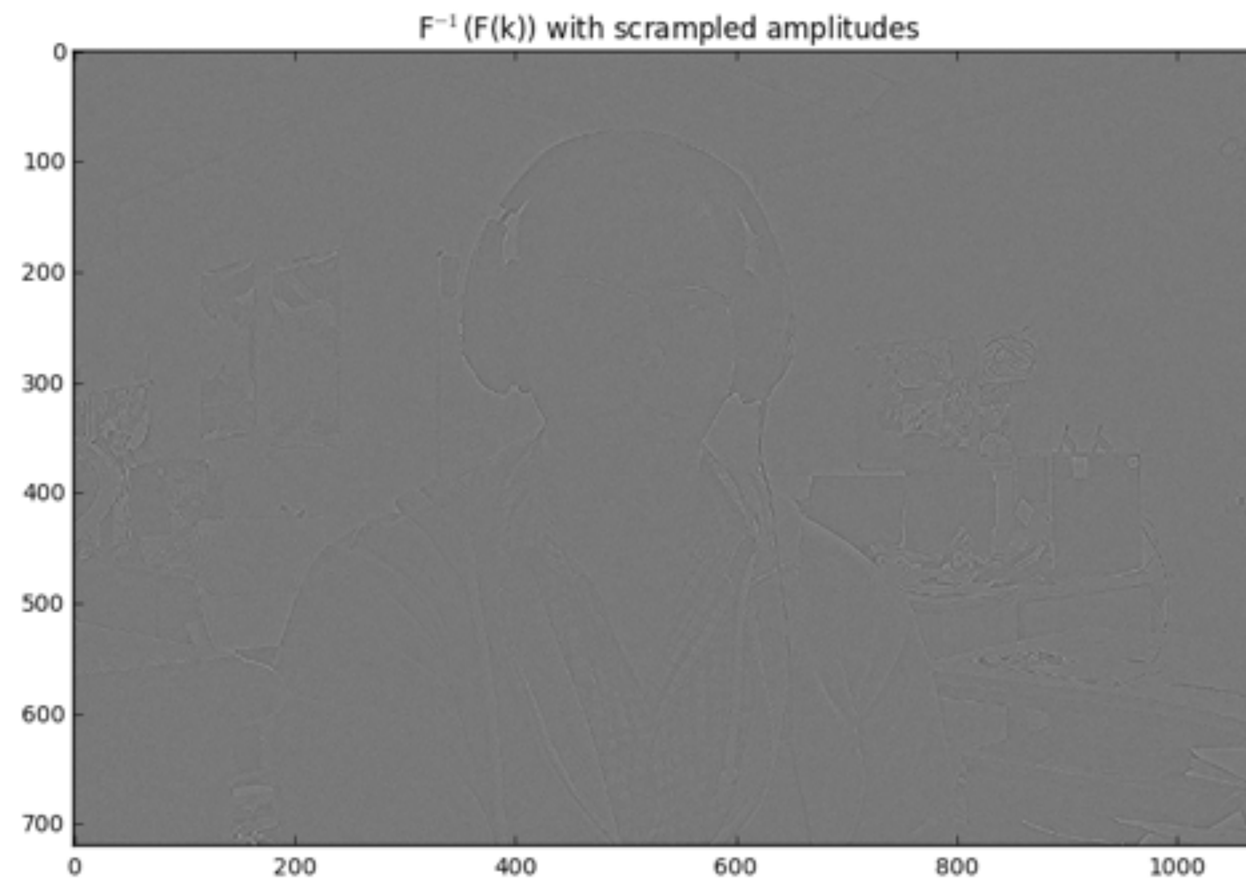


- Try giving this code an image of your own and see what the resulting amplitude and phase look like

Your own Fourier Transform

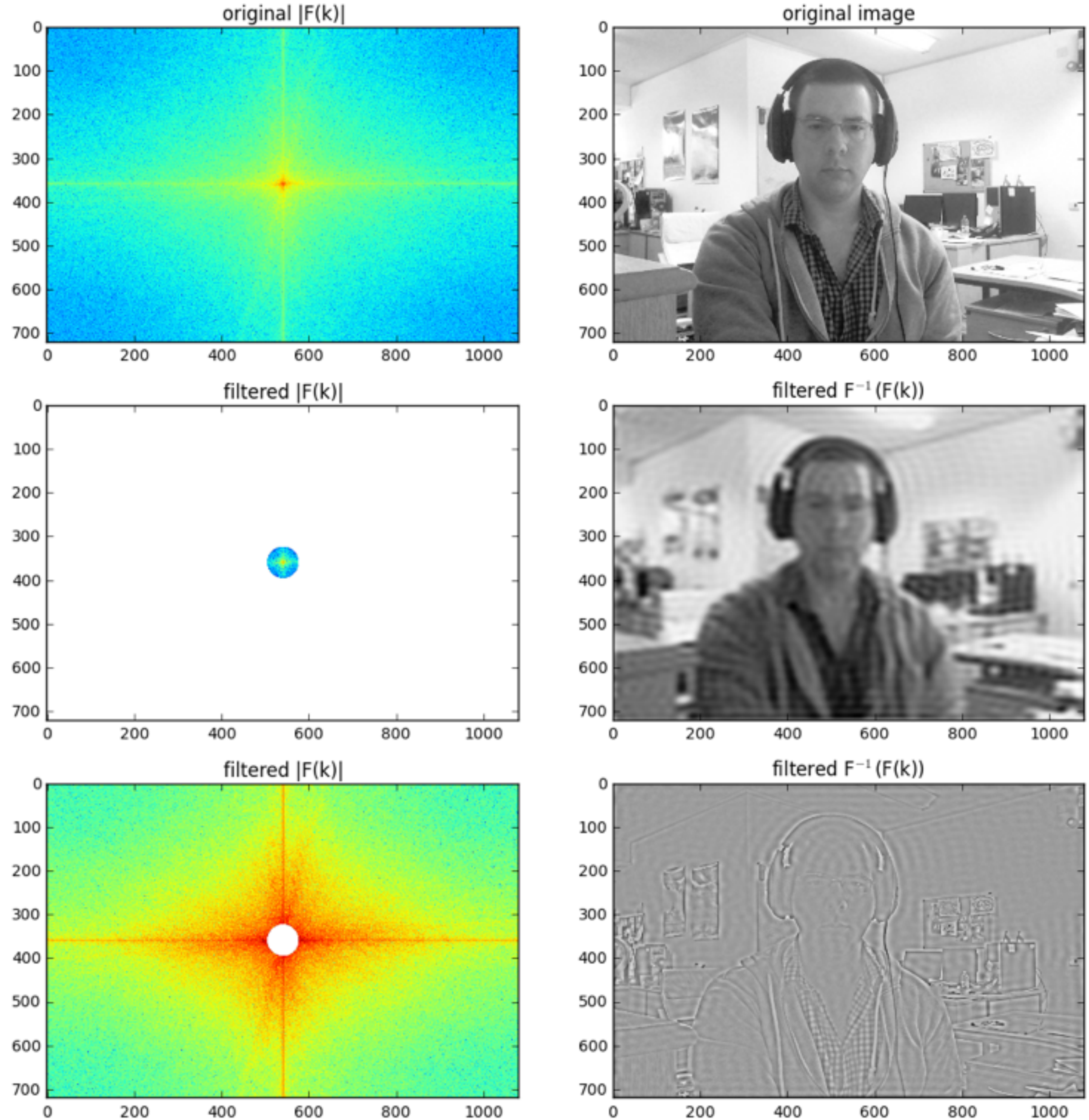
- It also produces images where the phase and amplitude has been randomly scrambled





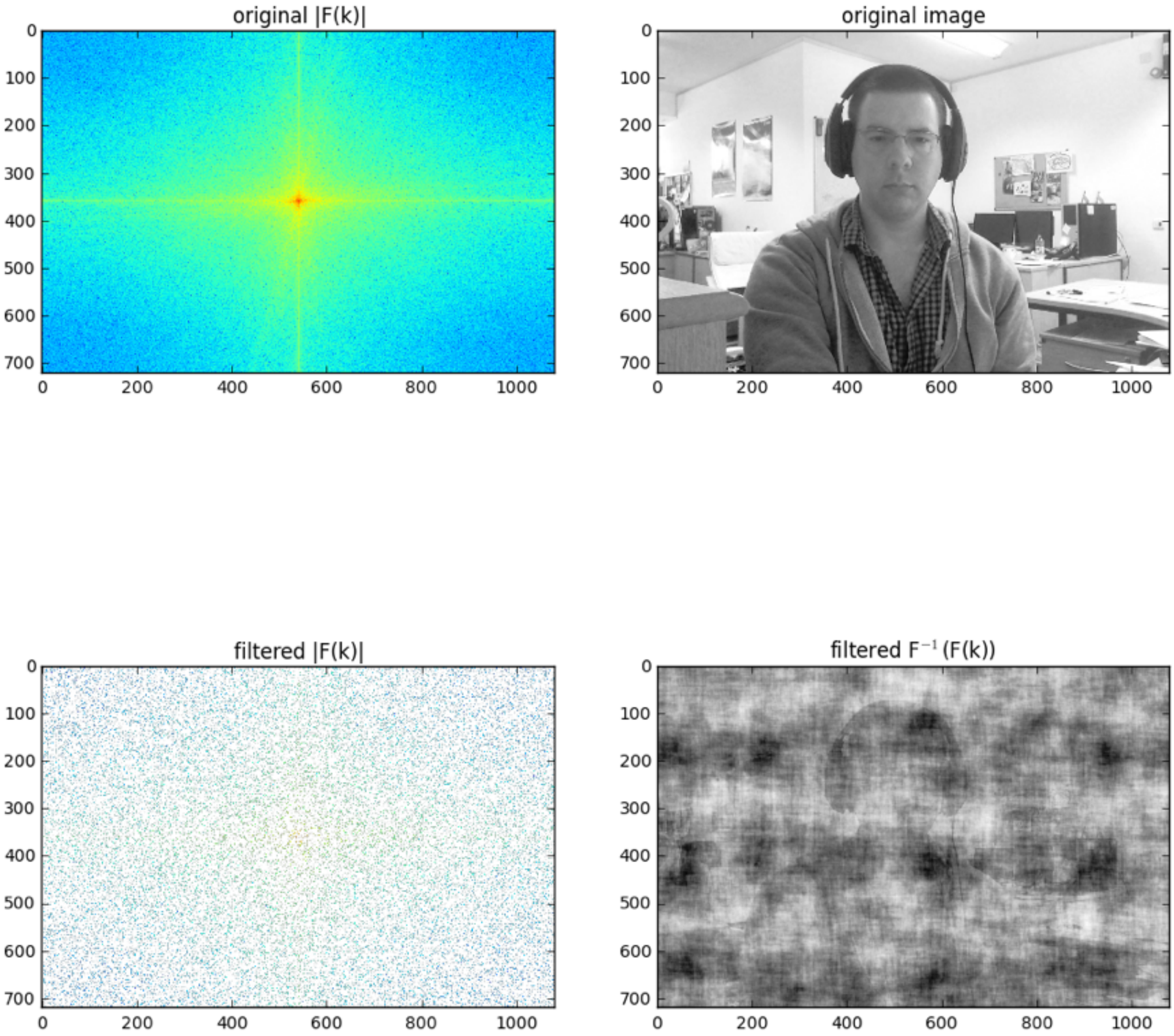
Filtering

- We can highlight the effect of spatial filtering
- Filtering out the long spatial scales results in a blurred image where the fine detail has been removed
- Conversely filtering out the short spatial scales results in an image where only the fine scale features exist


original |F(k)|
original image
filtered |F(k)|
filtered F⁻¹(F(k))
filtered |F(k)|
filtered F⁻¹(F(k))

Partial Sampling

- We can also demonstrate the effect of incomplete sampling of the *uv* plane
- The following images has had a random sample of *uv* points set to 0
- The resulting image is seriously degraded by the incomplete *uv* sampling


original |F(k)|
original image
filtered |F(k)|
filtered F⁻¹(F(k))

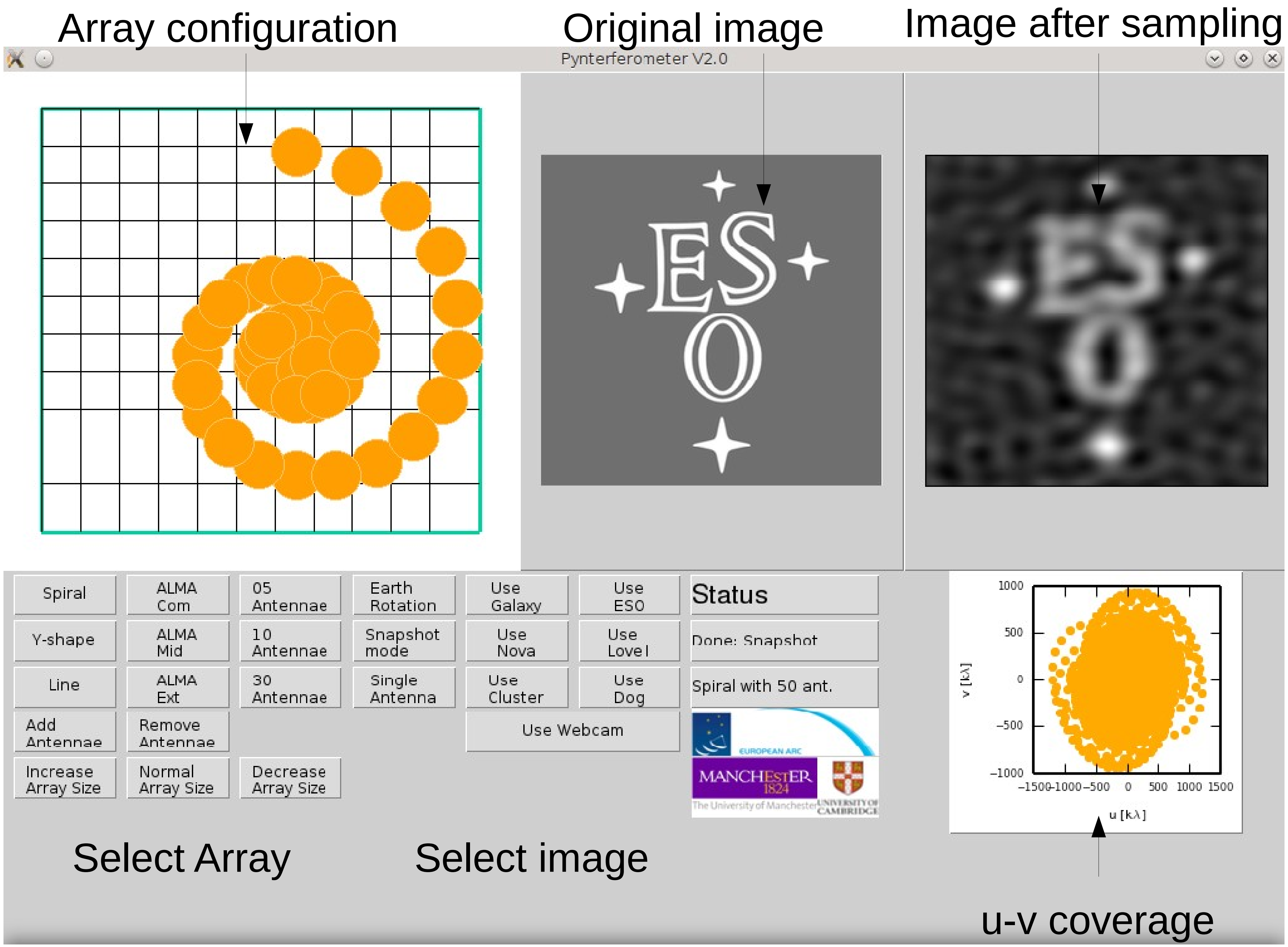
pynterferometer

- This program (written by Adam Alison and Sam George) shows you the results of observing an object with a variety of array configurations

- There is no set script to follow: experiment with different configurations to get an intuitive feel for how well they can reproduce an image

- To start, cd to the directory where you have installed the package and type:

```
python Pyntv2ERIS.py
```


Array configuration
Original image
Image after sampling
Pynterferometer V2.0
Spiral
ALMA Com
05 Antennae
Earth Rotation
Use Galaxy
Use ESO
Status
Y-shape
ALMA Mid
10 Antennae
Snapshot mode
Use Nova
Use Lovel
Done: Snapshot
Line
ALMA Ext
30 Antennae
Single Antenna
Use Cluster
Use Dog
Spiral with 50 ant.
Add Antennae
Remove Antennae
Use Webcam
Increase Array Size
Normal Array Size
Decrease Array Size
v [kλ]
u [kλ]
Select Array
Select image
u-v coverage

Things to try

- Select your favourite object
- Start with 5 antenna linear array
    - Remove all but 2 antennas (single baseline)
    - Change the spacing (increase/decrease array size)
- Add antennas
- Turn on Earth rotation
- Look at other configurations
    - Y for VLA
    - ALMA
- What happens when you make the array too large or too small?