Friedrich-Alexander-Universität
Technische Fakultät

Lecture 3, Part 4

Thinking in The Frequency Domain 2/2

Computer Vision
Summer Semester 2022

Prof. Bernhard Egger, Prof. Tim Weyrich, Prof. Andreas Maier

Frequencies

- Fourier series and Fourier transform
- Filtering in frequency domain
- Deconvolution
- JPEG compression

Fourier series

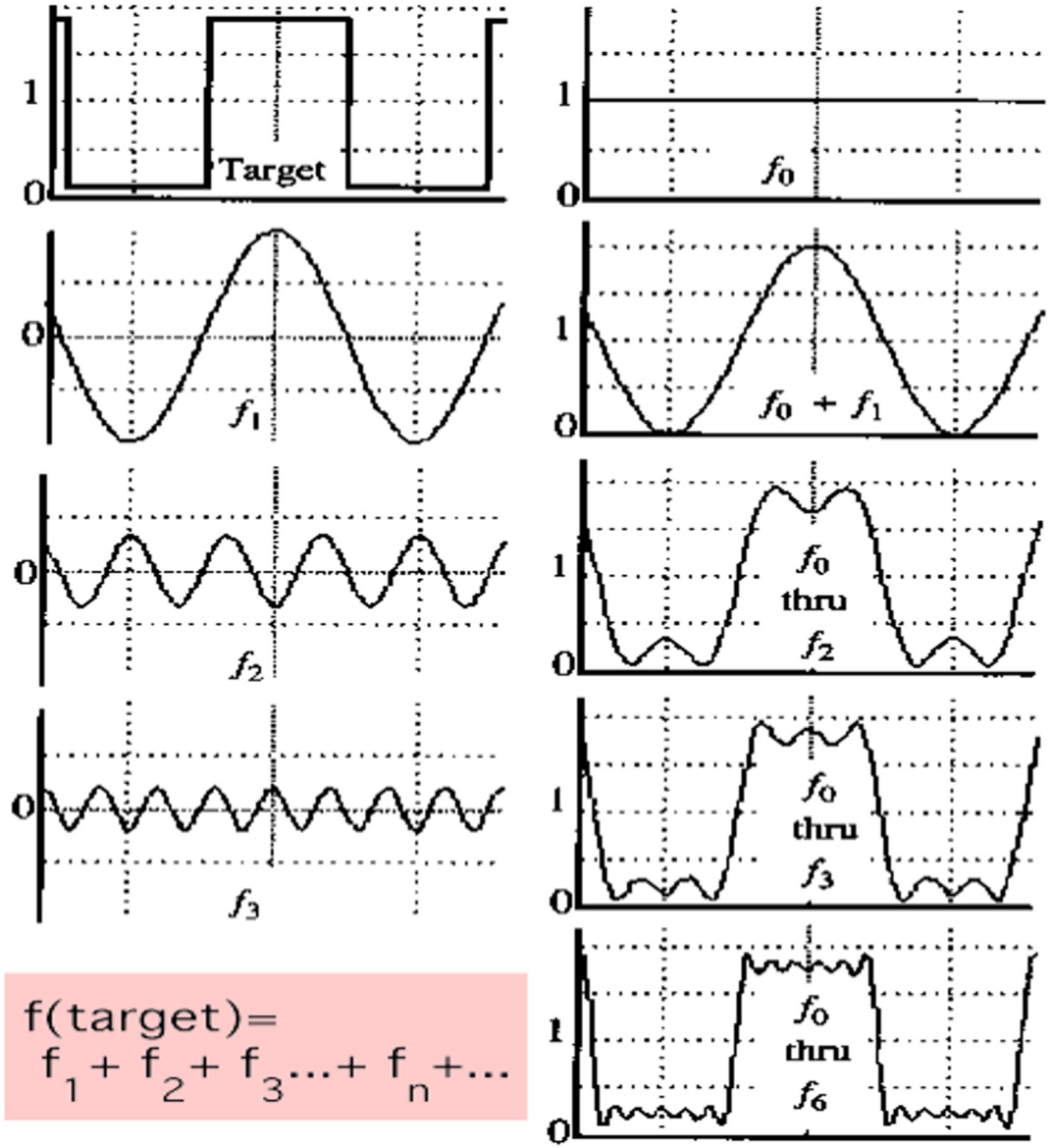
Idea (1807):

> ***Any*** univariate function can be rewritten as a weighted sum of sines and cosines of different frequencies.

Our building block:

$$A\sin(\omega t) + B\cos(\omega t)$$

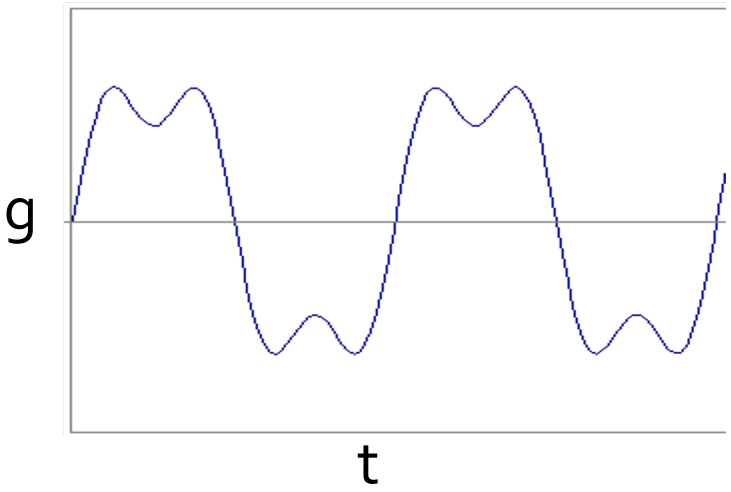
Add enough of them to get any signal *g(t)* you want!



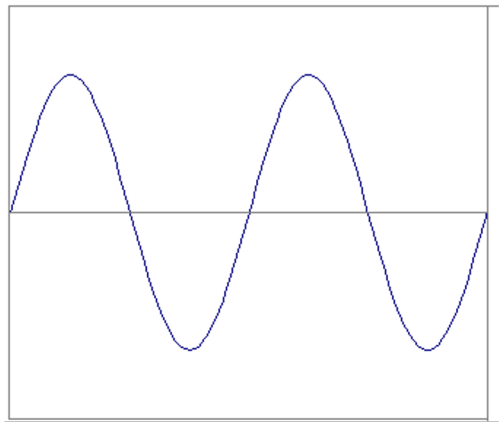
James Hays

$$t = [0,2], f = 1$$

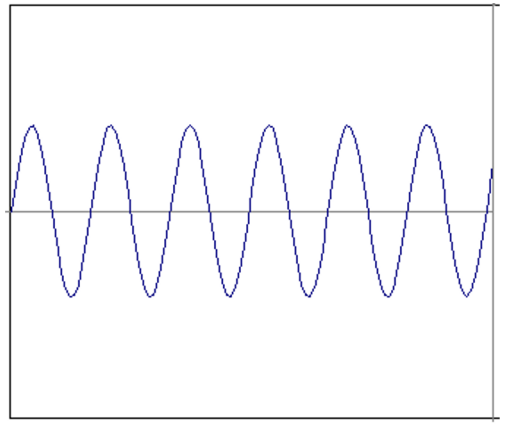
$$g(t) = \sin(2\pi f\, t) + (1/3)\sin(2\pi(3f)t)$$

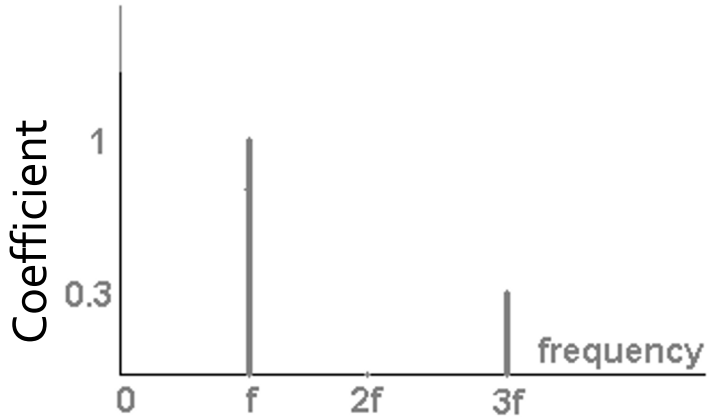


=



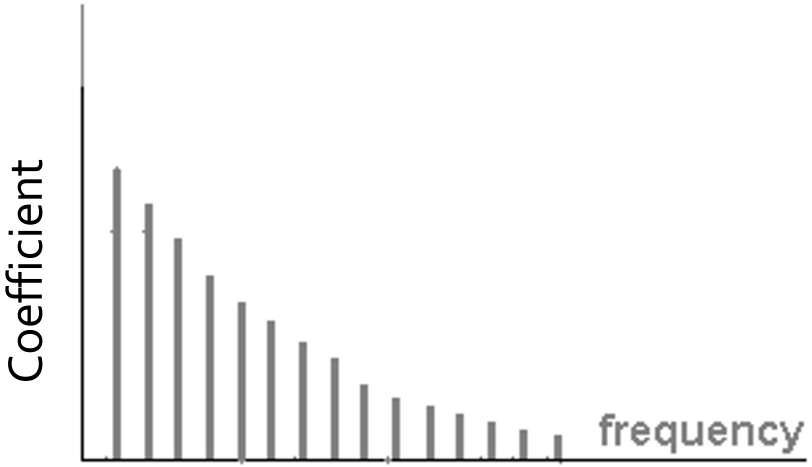
+





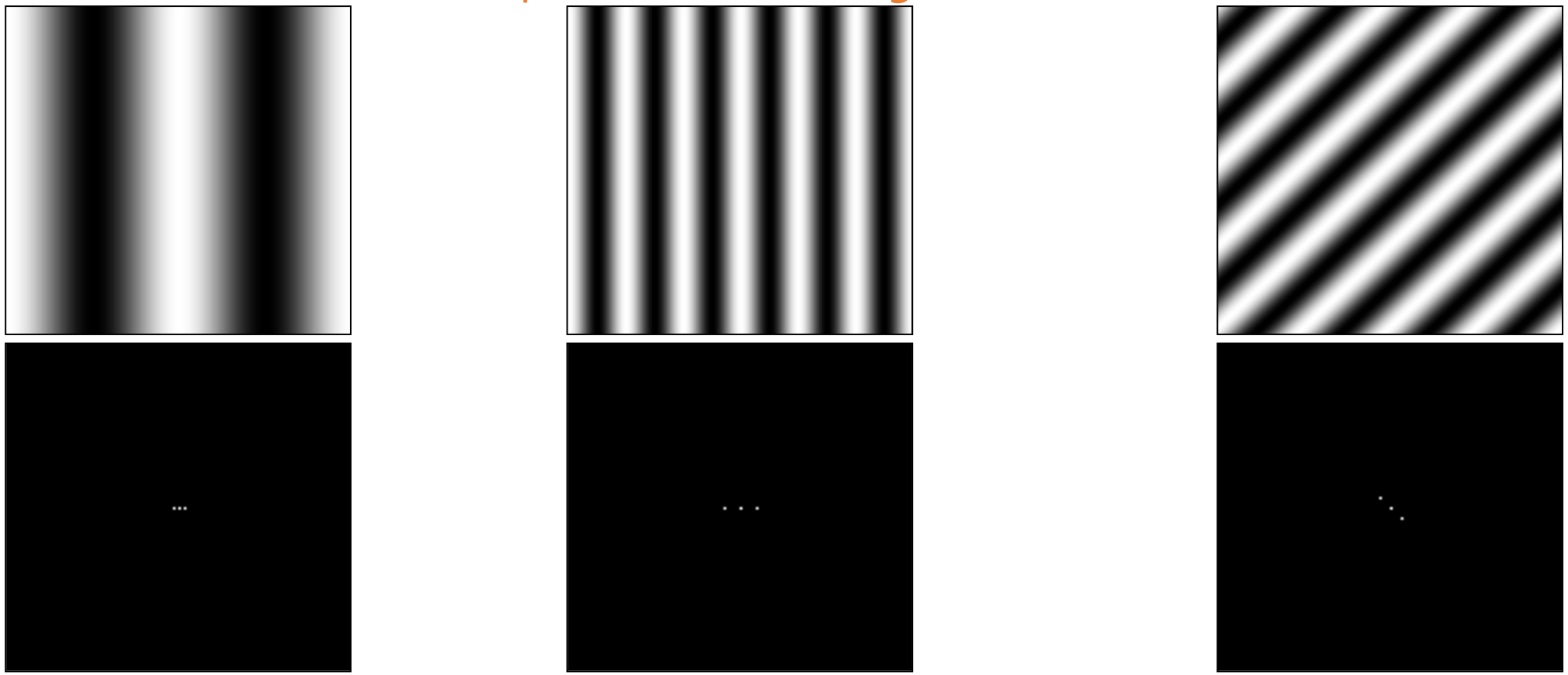
Square wave spectra

$$= A \sum_{f=1}^{\infty} \frac{1}{f} \sin(2\pi f t)$$



Fourier analysis in images

Spatial domain images

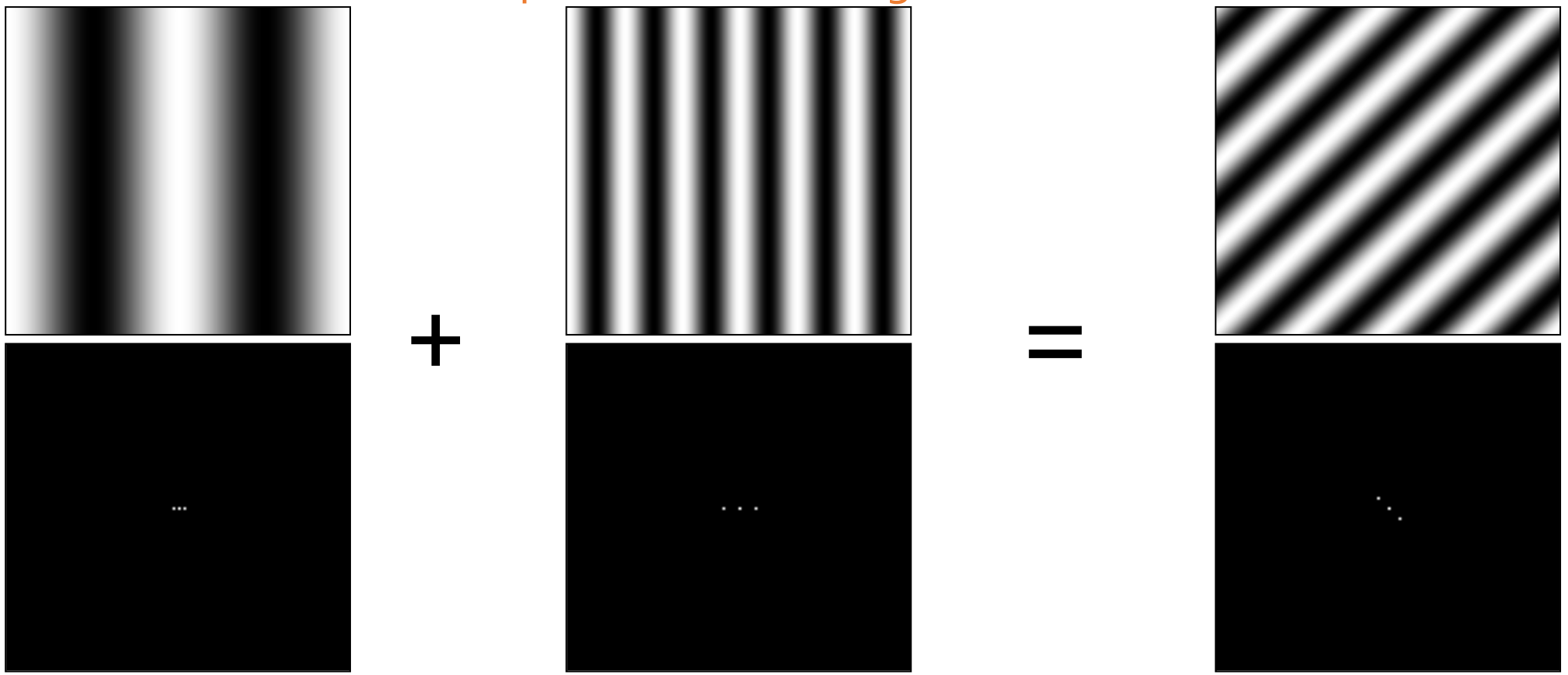


Fourier decomposition frequency amplitude images

More:http://www.cs.unm.edu/~brayer/vision/fourier.html

Signals can be composed

Spatial domain images



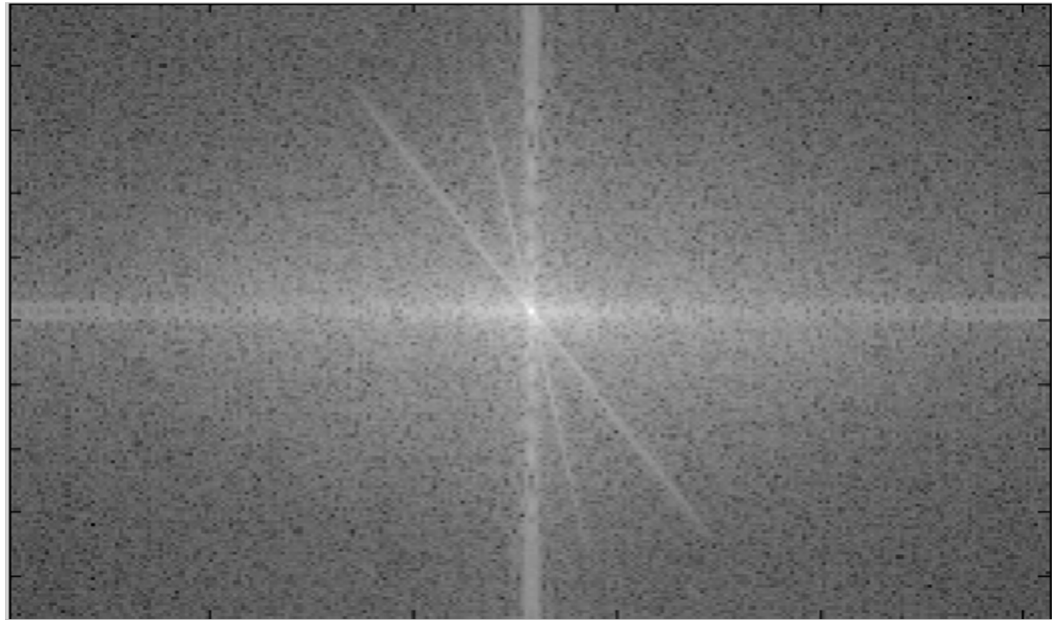
Fourier decomposition frequency amplitude images

http://sharp.bu.edu/~slehar/fourier/fourier.html#filtering More:http://www.cs.unm.edu/~brayer/vision/fourier.html

Natural image

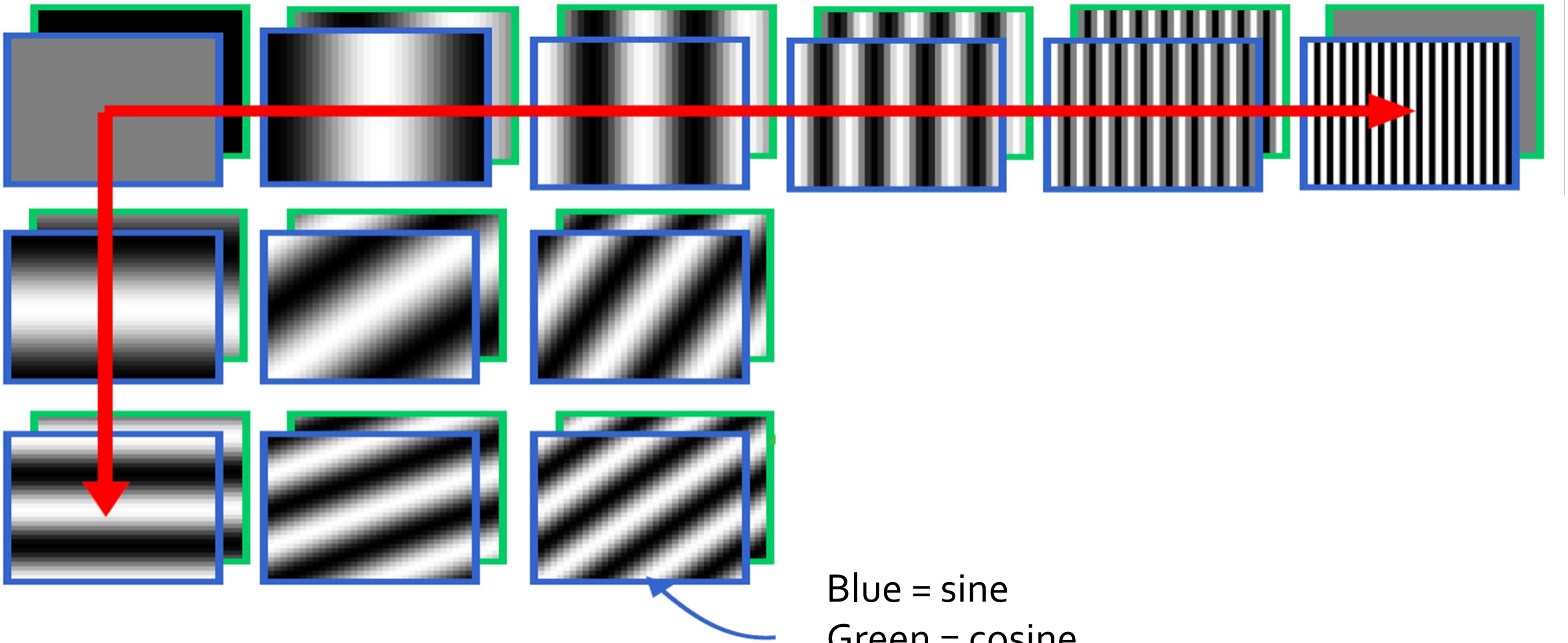
Natural image

Fourier decomposition
Frequency coefficients (amplitude)



What does it mean to be at pixel x,y?
What does it mean to be more or less bright in the Fourier decomposition image?

Fourier Bases



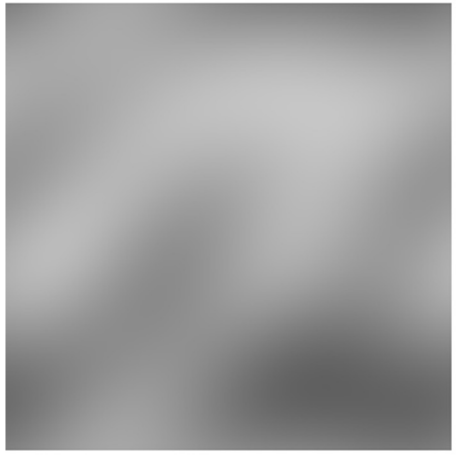
James Hays

Basis reconstruction

Full image

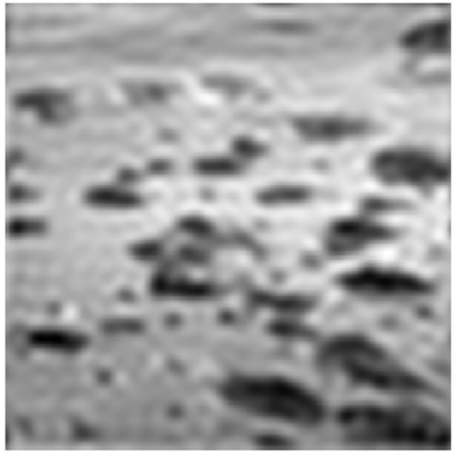
First 1 basis fn

First 4 basis fns



First 9 basis fns

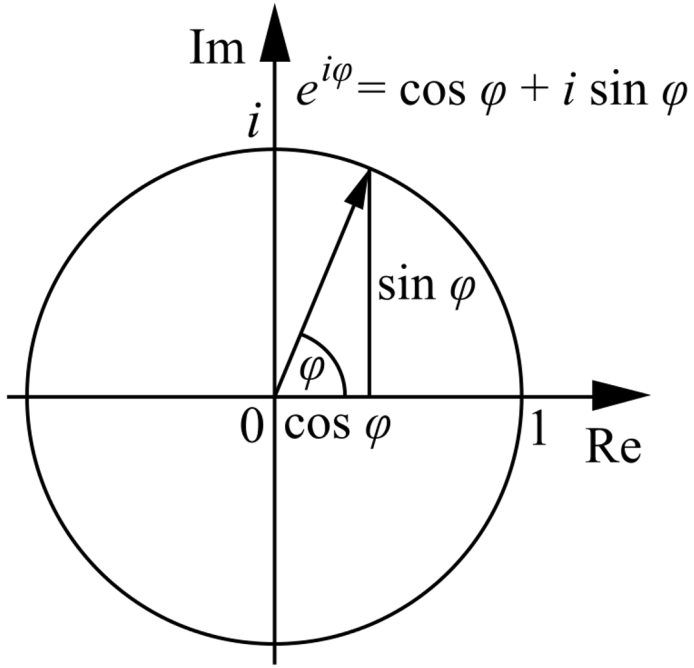
First 16 basis fns



First 400 basis fns

Fourier Transform

- Stores the amplitude and phase at each frequency:
  - For mathematical convenience, this is often notated in terms of real and complex numbers
  - Related by Euler's formula



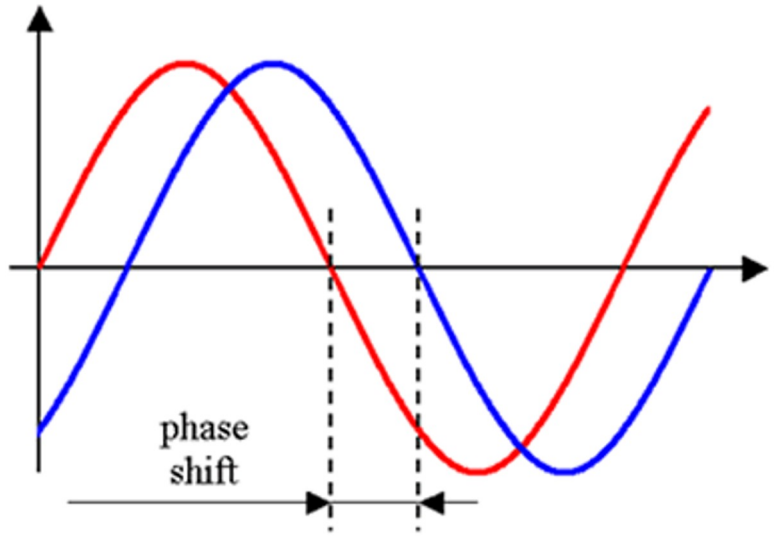
- Amplitude encodes how much signal there is at a particular frequency:

$$A = \pm\sqrt{\mathrm{Re}(\varphi)^2 + \mathrm{Im}(\varphi)^2}$$

- Phase encodes spatial information (indirectly):

$$\phi = \tan^{-1}\frac{\mathrm{Im}(\varphi)}{\mathrm{Re}(\varphi)}$$

Amplitude / Phase



- Amplitude tells you “how much”
- Phase tells you “where”

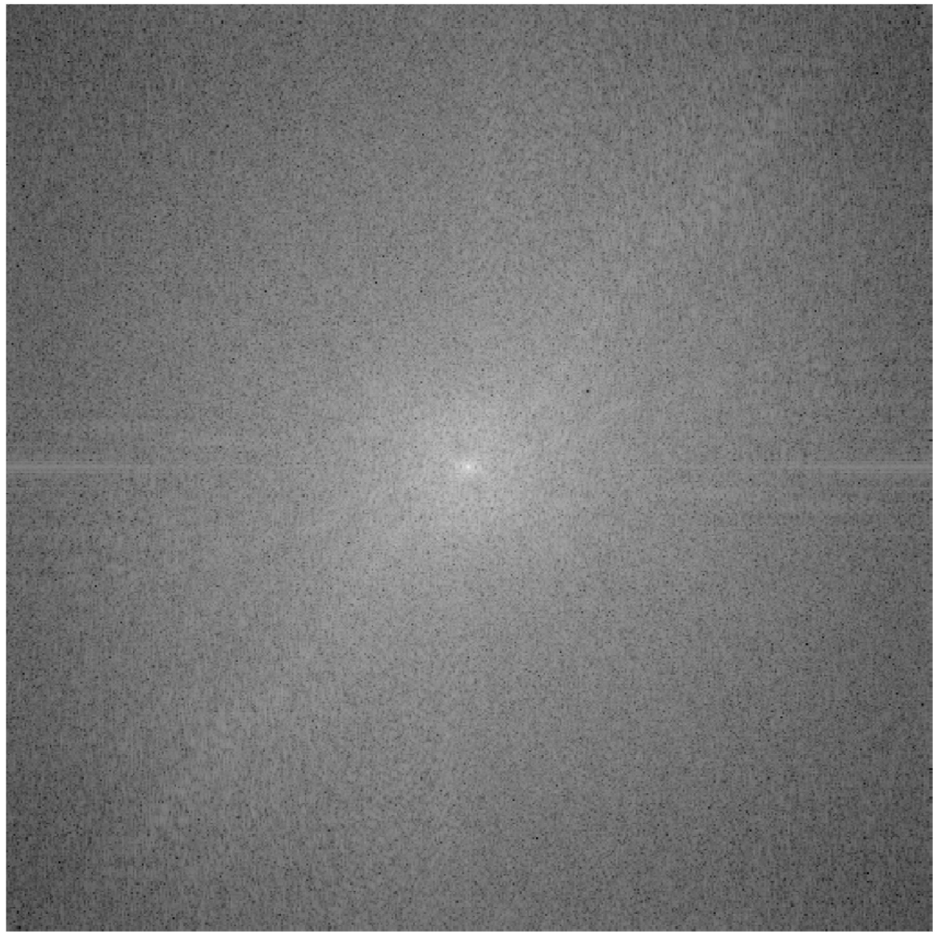
- Translate the image?
  - Amplitude unchanged
  - Adds a constant to the phase.

Amplitude vs. phase demo: cheetah

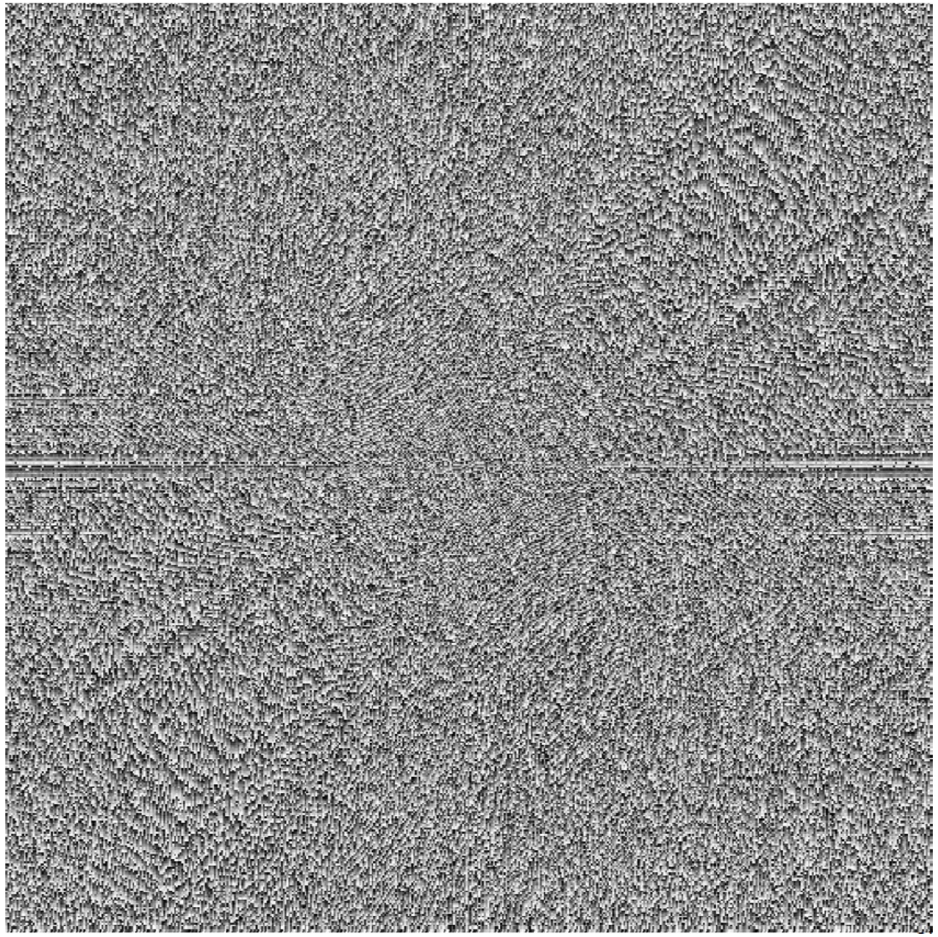
Efros

Amplitude vs. phase demo: cheetah

Amplitude



Phase



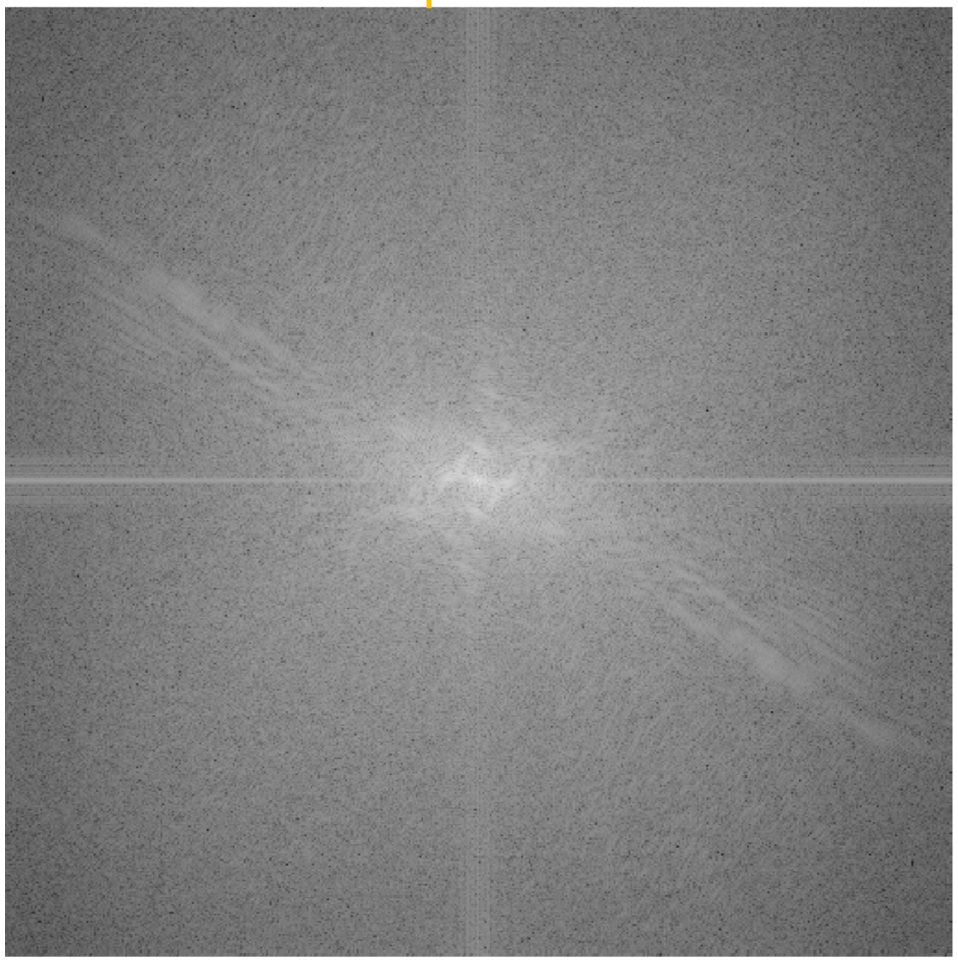
Efros

Amplitude vs. phase demo: zebra

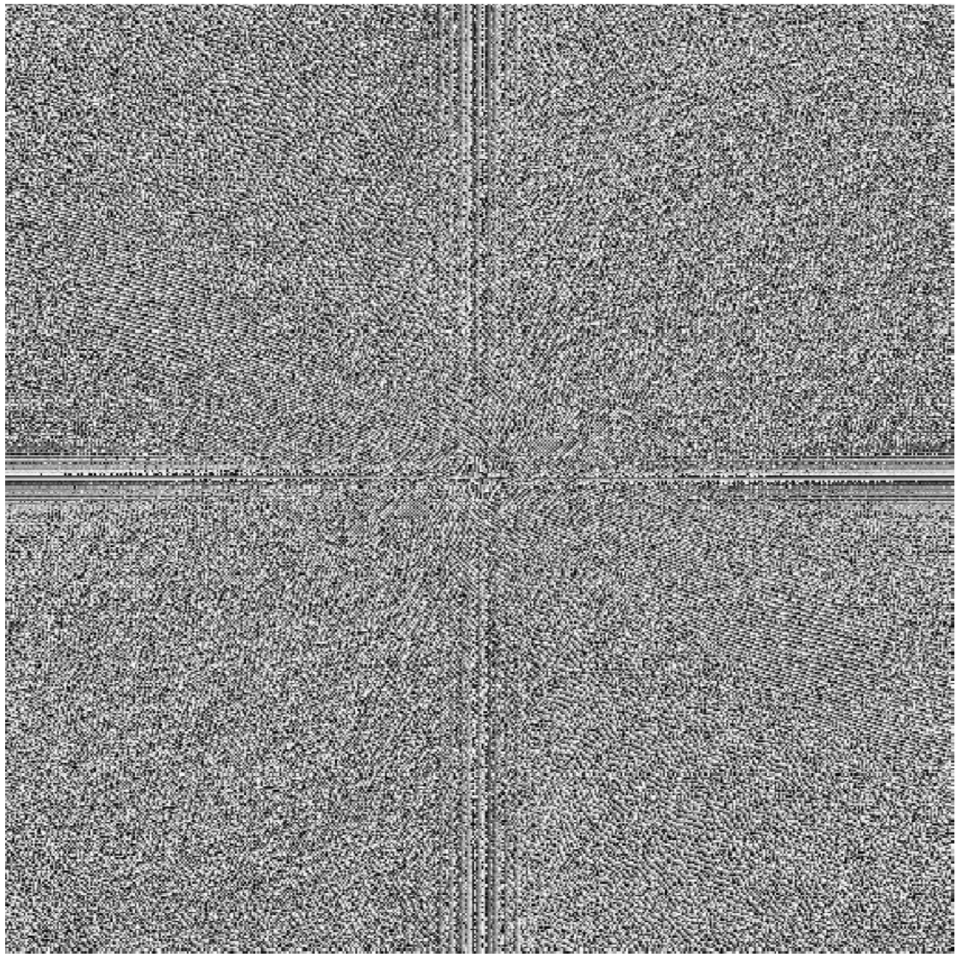
Efros

Amplitude vs. phase demo: zebra

Amplitude



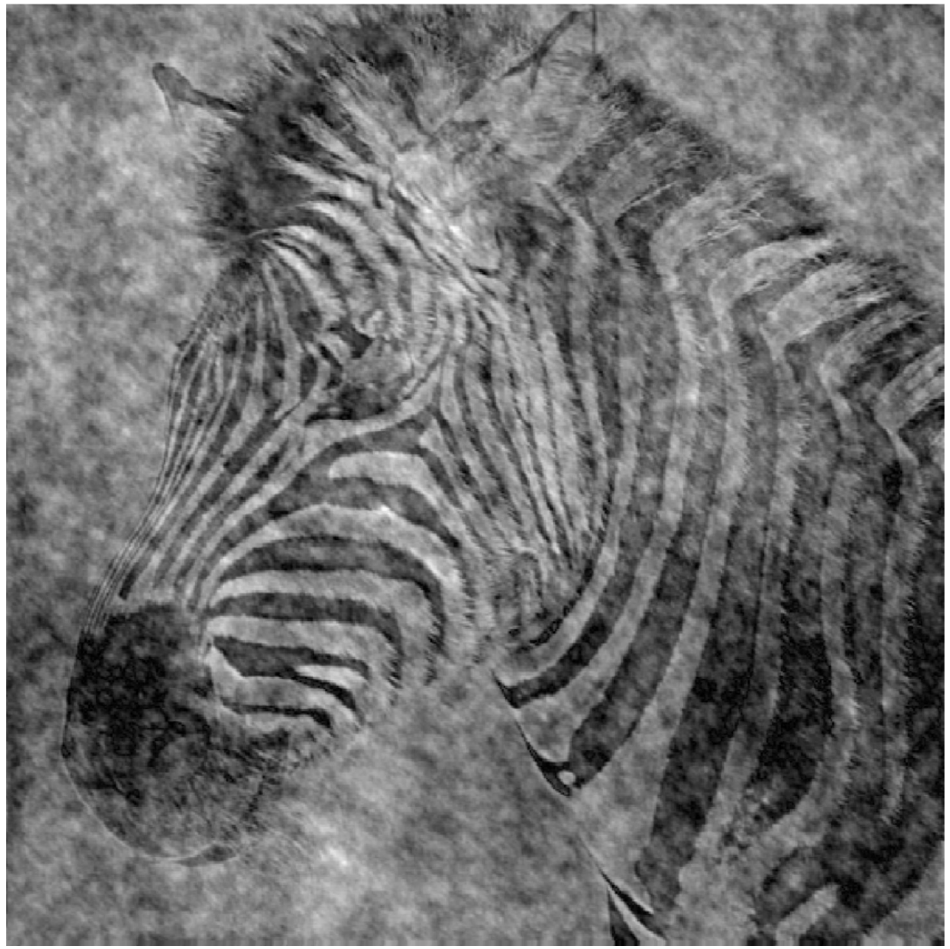
Phase



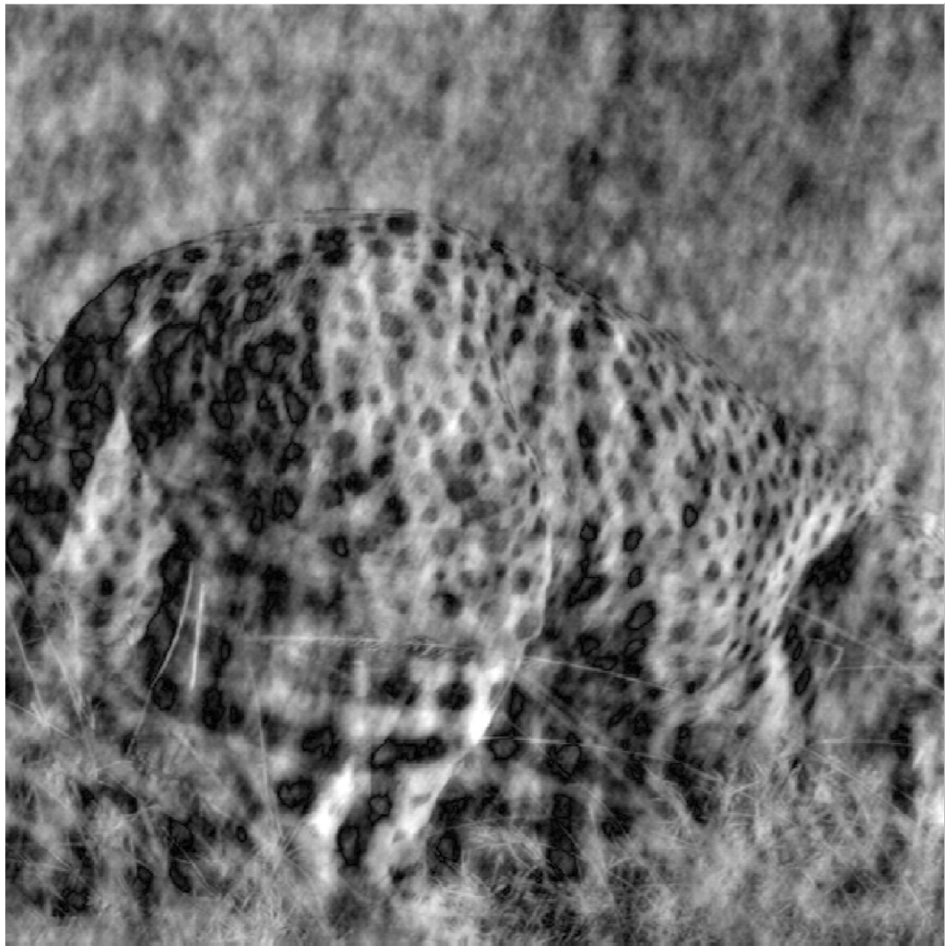
Efros

Is the image info in amplitude or phase?

Zebra phase, cheetah amplitude



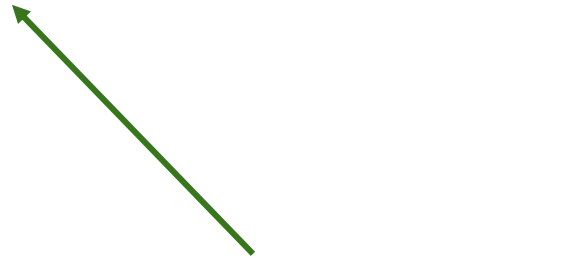
Cheetah phase, zebra amplitude



Efros

Amplitude vs. phase

- The frequency amplitude of natural images are quite similar
  - Heavy in low frequencies, falling off in high frequencies
  - Will *any* image be like that, or is it a property of the world we live in?
- Most information in the image is carried in the phase, not the amplitude
  - Not quite clear why



'Open' Research Area!

Properties of Fourier Transforms

- Linearity: $\mathrm{F}[ax(t)+by(t)] = a\,\mathrm{F}[x(t)] + b\,\mathrm{F}[y(t)]$
- Fourier transform of a real signal is symmetric about the origin
- The energy of the signal is the same as the energy of its Fourier transform

The Convolution Theorem

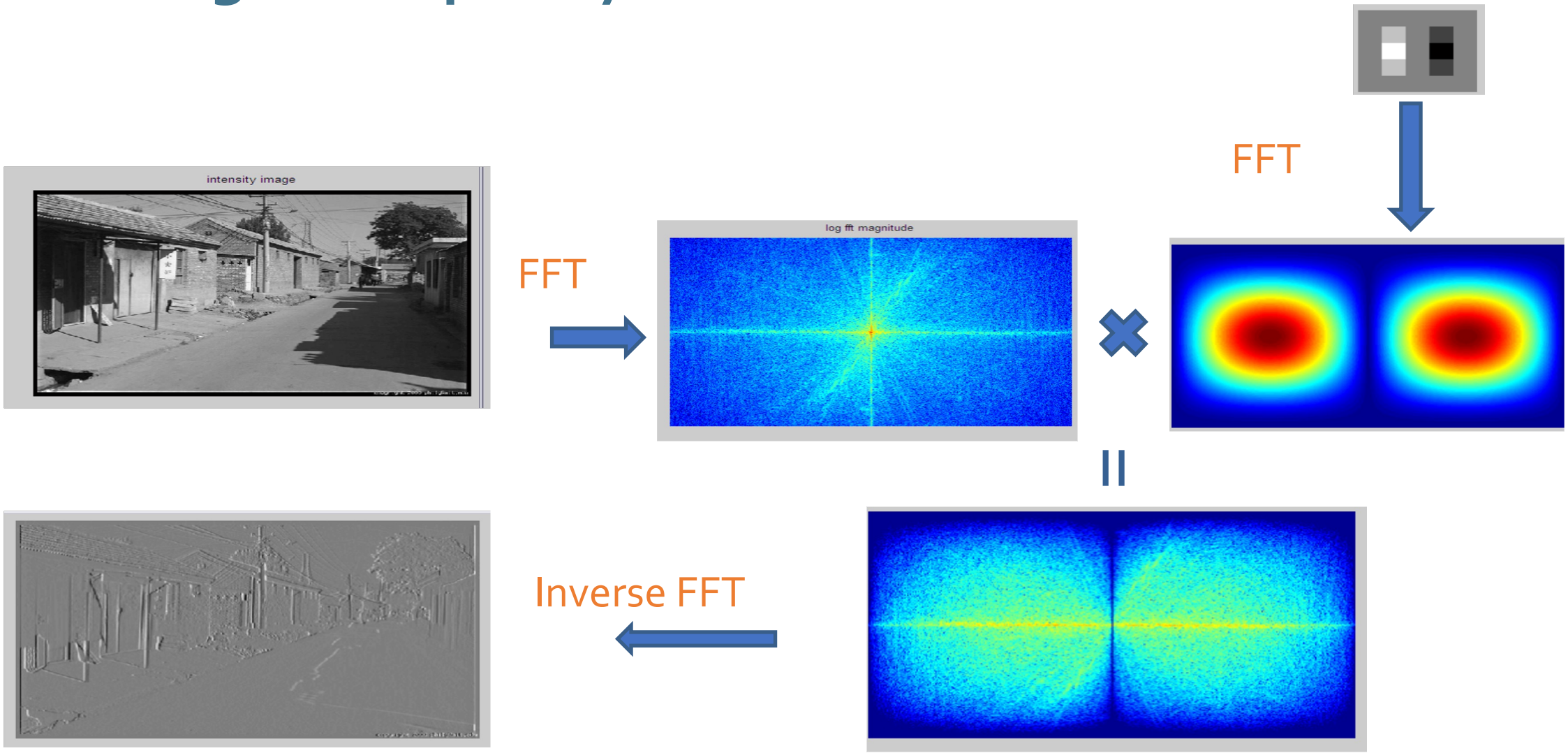
- The Fourier transform of the convolution of two functions is the product of their Fourier transforms

$$\mathrm{F}[g * h] = \mathrm{F}[g]\mathrm{F}[h]$$

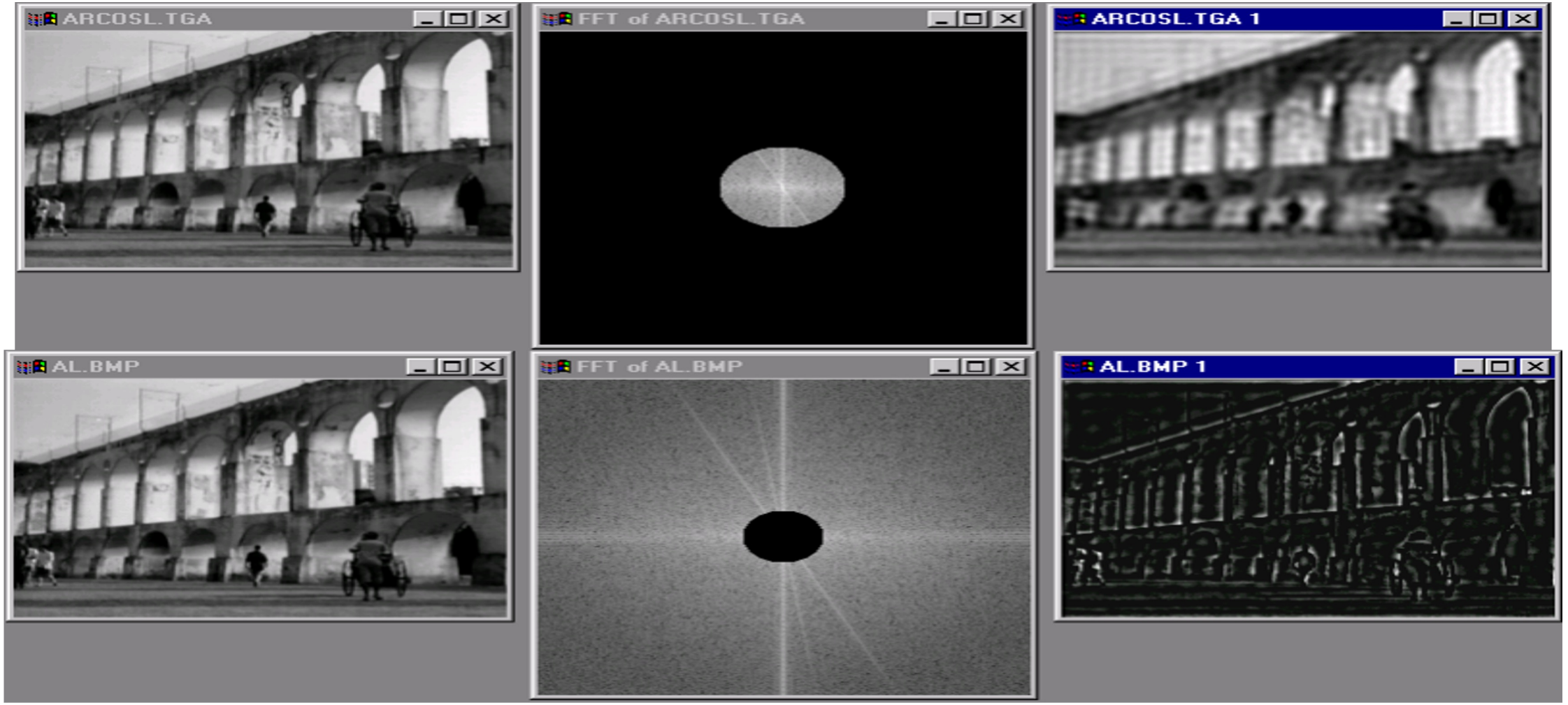
- **Convolution** in spatial domain is equivalent to **multiplication** in frequency domain!

$$g * h = \mathrm{F}^{-1}[\mathrm{F}[g]\mathrm{F}[h]]$$

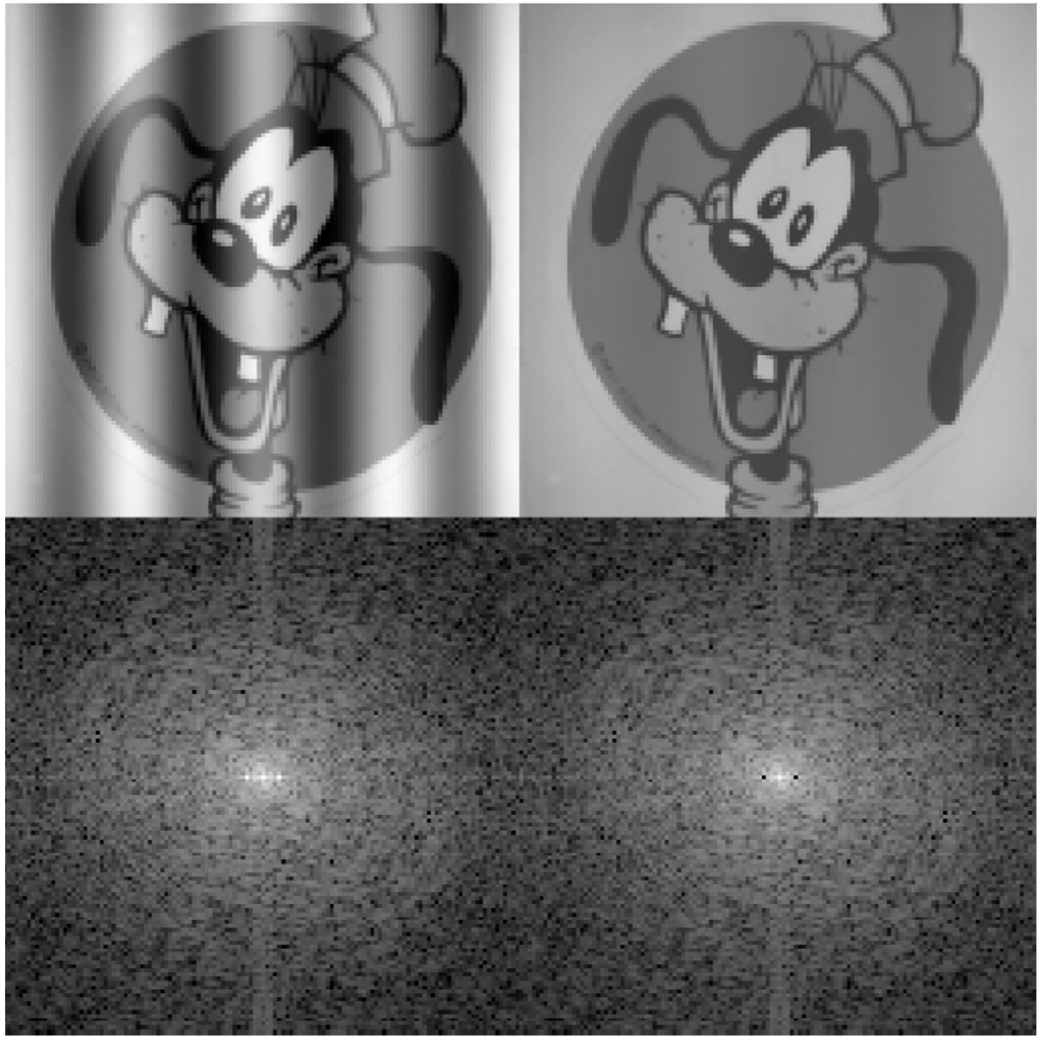
Filtering in Frequency Domain



Low and High Pass filtering



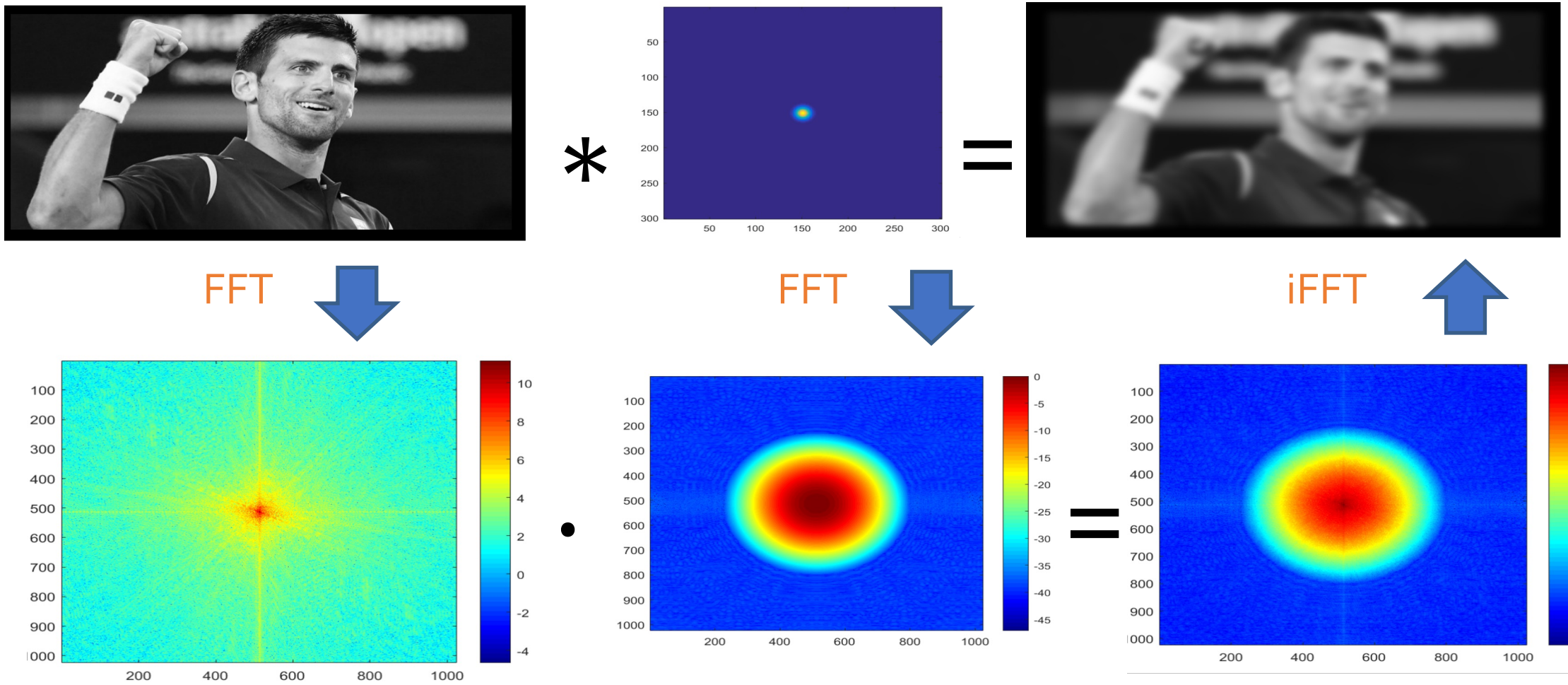
Removing frequency bands



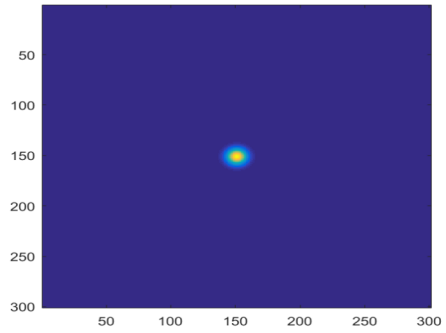
Is convolution invertible?

- If convolution is just multiplication in the Fourier domain, isn't deconvolution just division?
- Sometimes, it clearly is invertible (e.g. a convolution with an identity filter)
- In one case, it clearly isn't invertible (e.g. convolution with an all zero filter)
- What about for common filters like a Gaussian?

Convolution



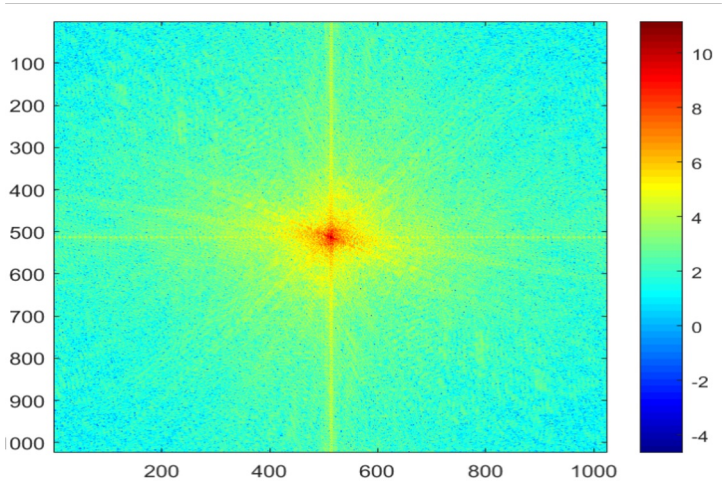
Deconvolution?



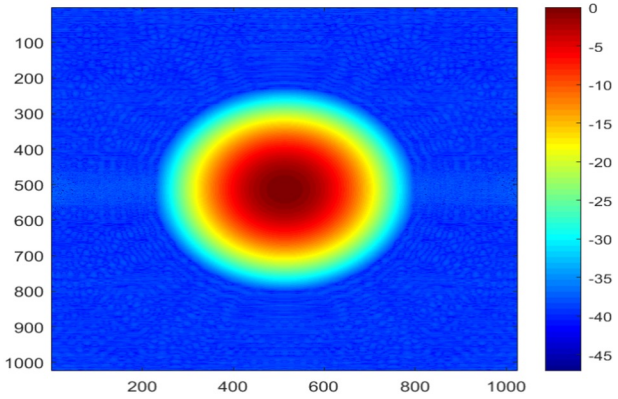
iFFT

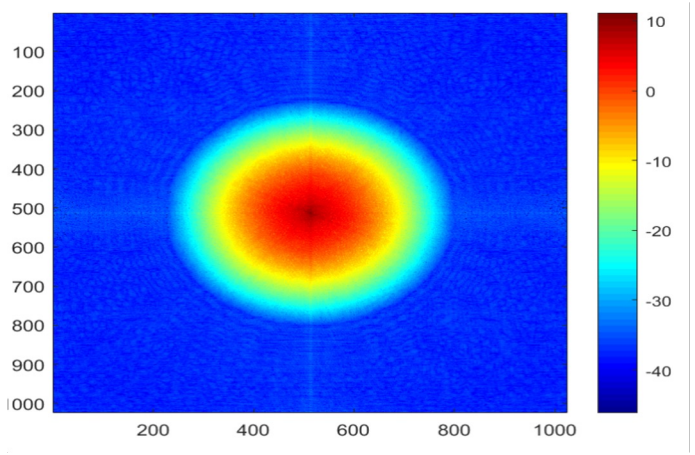
FFT

FFT



=

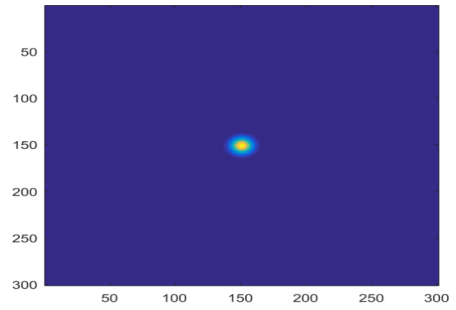




James Hays

But under more realistic conditions

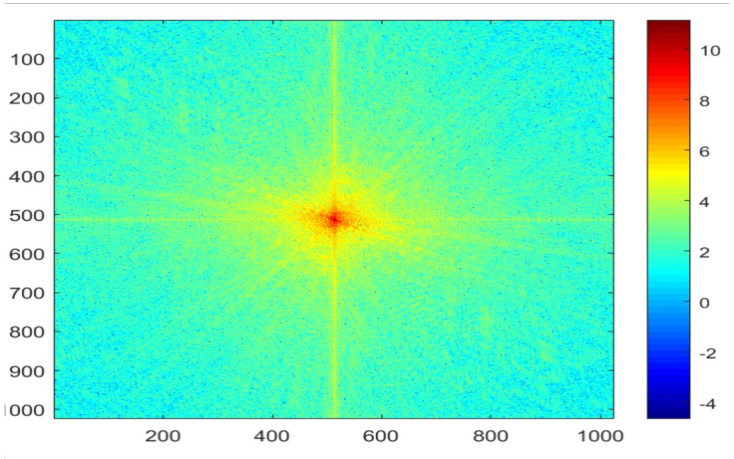
Random noise, .001 magnitude



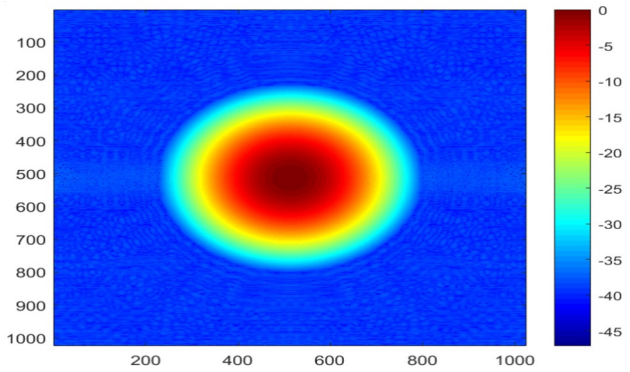
iFFT

FFT

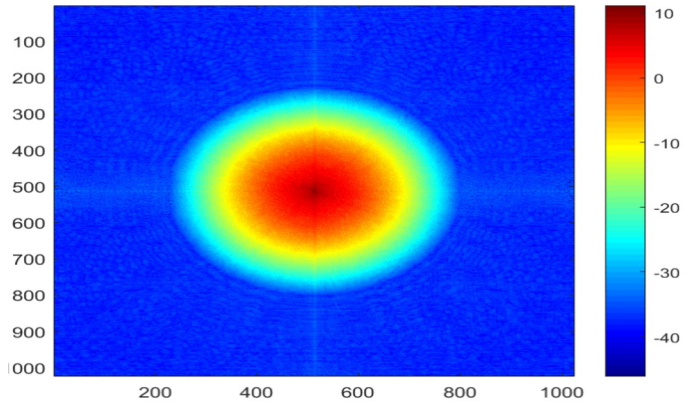
FFT



=



▪
▪



James Hays

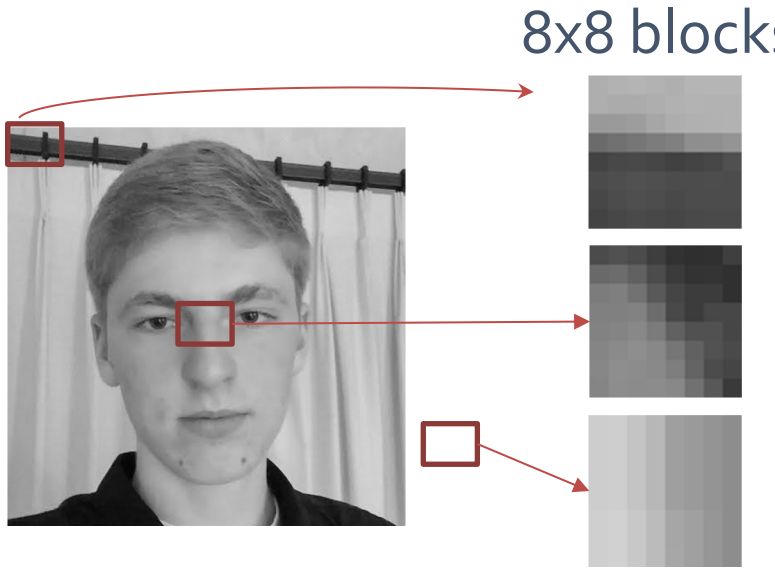
Deconvolution is hard.

- Active research area.
- Even if you know the filter (non-blind deconvolution), it is still hard and requires strong *regularization* to counteract noise.
- If you don’t know the filter (blind deconvolution), then it is harder still.

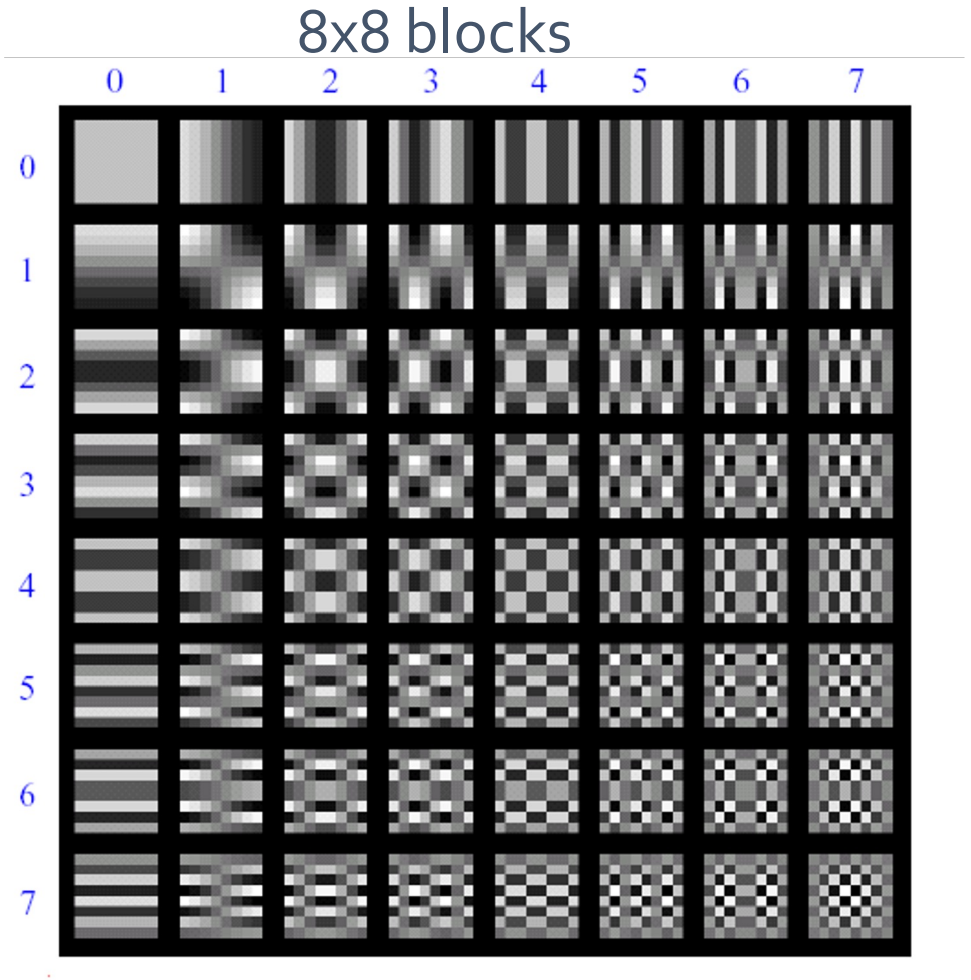
Thinking in Frequency - Compression

How is it that a 4MP image can be compressed to a few hundred KB without a noticeable change?

Lossy Image Compression (JPEG)



- The first coefficient $B(0,0)$ is the DC component, the average intensity
- The top-left coeffs represent low frequencies, the bottom right represent high frequencies



Block-based Discrete Cosine Transform (DCT)

Efros

Image compression using DCT

- Compute DCT filter responses in each 8x8 block
- Quantize to integer (div. by magic number; round)
  - More coarsely for high frequencies (which also tend to have smaller values)
  - Many quantized high frequency values will be zero

Filter responses

$$G = \begin{bmatrix} -415.38 & -30.19 & -61.20 & 27.24 & 56.13 & -20.10 & -2.39 & 0.46 \\ 4.47 & -21.86 & -60.76 & 10.25 & 13.15 & -7.09 & -8.54 & 4.88 \\ -46.83 & 7.37 & 77.13 & -24.56 & -28.91 & 9.93 & 5.42 & -5.65 \\ -48.53 & 12.07 & 34.10 & -14.76 & -10.24 & 6.30 & 1.83 & 1.95 \\ 12.12 & -6.55 & -13.20 & -3.95 & -1.88 & 1.75 & -2.79 & 3.14 \\ -7.73 & 2.91 & 2.38 & -5.94 & -2.38 & 0.94 & 4.30 & 1.85 \\ -1.03 & 0.18 & 0.42 & -2.42 & -0.88 & -3.02 & 4.12 & -0.66 \\ -0.17 & 0.14 & -1.07 & -4.19 & -1.17 & -0.10 & 0.50 & 1.68 \end{bmatrix}$$

($u \rightarrow$, $v \downarrow$)

Quantization divisers (element-wise)

$$Q = \begin{bmatrix} 16 & 11 & 10 & 16 & 24 & 40 & 51 & 61 \\ 12 & 12 & 14 & 19 & 26 & 58 & 60 & 55 \\ 14 & 13 & 16 & 24 & 40 & 57 & 69 & 56 \\ 14 & 17 & 22 & 29 & 51 & 87 & 80 & 62 \\ 18 & 22 & 37 & 56 & 68 & 109 & 103 & 77 \\ 24 & 35 & 55 & 64 & 81 & 104 & 113 & 92 \\ 49 & 64 & 78 & 87 & 103 & 121 & 120 & 101 \\ 72 & 92 & 95 & 98 & 112 & 100 & 103 & 99 \end{bmatrix}$$

Quantized values

$$B = \begin{bmatrix} -26 & -3 & -6 & 2 & 2 & -1 & 0 & 0 \\ 0 & -2 & -4 & 1 & 1 & 0 & 0 & 0 \\ -3 & 1 & 5 & -1 & -1 & 0 & 0 & 0 \\ -3 & 1 & 2 & -1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

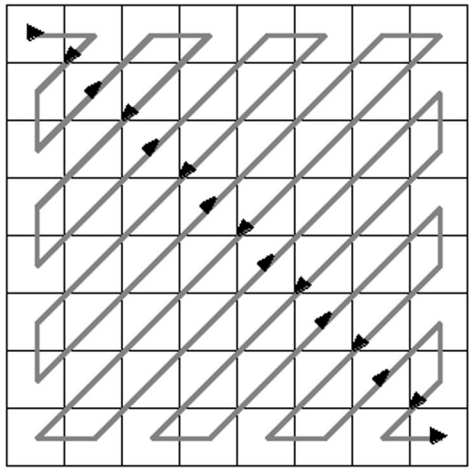
JPEG Encoding

- Entropy coding (Huffman-variant)

Quantized values

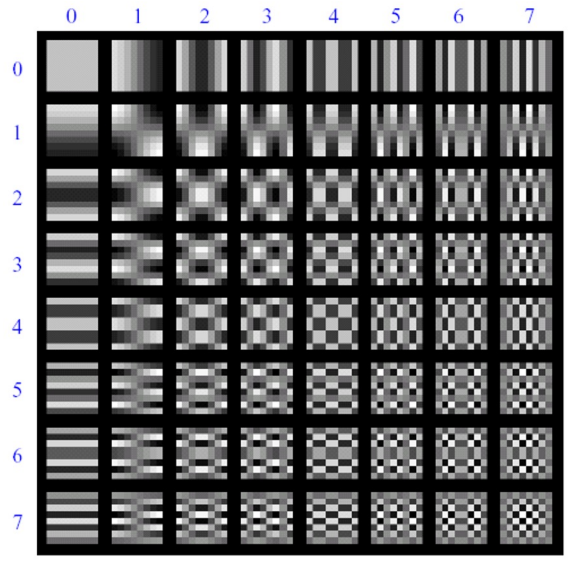
$$B = \begin{bmatrix} -26 & -3 & -6 & 2 & 2 & -1 & 0 & 0 \\ 0 & -2 & -4 & 1 & 1 & 0 & 0 & 0 \\ -3 & 1 & 5 & -1 & -1 & 0 & 0 & 0 \\ -3 & 1 & 2 & -1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Linearize B like this.



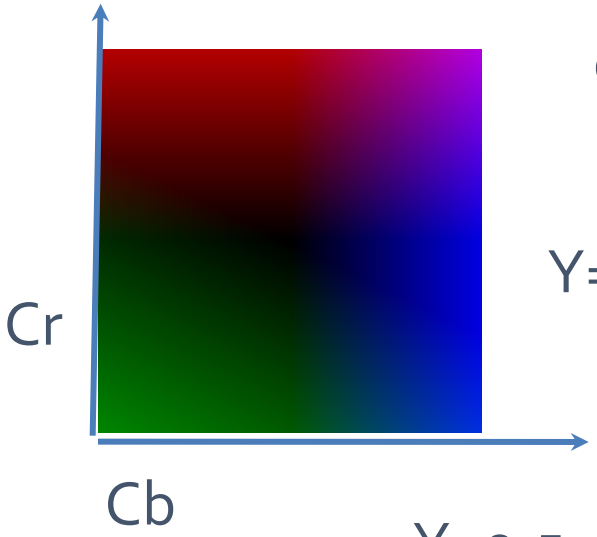
Helps compression:

- We throw away the high frequencies (‘o’).
- The zig zag pattern increases in frequency space, so long runs of zeros.

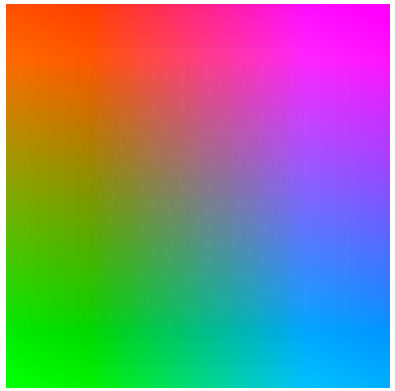


Color spaces: YCbCr

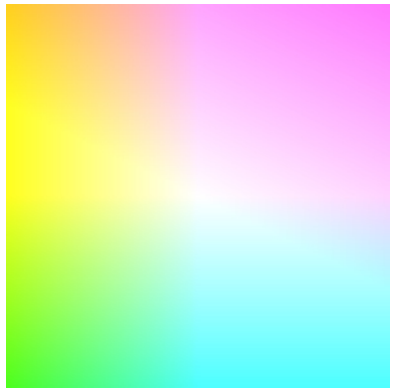
Fast to compute, good for compression, used by TV



Y=0.5



Y=1



Y

(Cb=0.5,Cr=0.5)

Cb

(Y=0.5,Cr=0.5)

Cr

(Y=0.5,Cb=05)

James Hays

Most JPEG images & videos subsample chroma

PSP Comp 3
2x2 Chroma subsampling
285K

Original
1,261K lossless
968K PNG

JPEG Compression Summary

1. Convert image to YCrCb
2. Subsample color by factor of 2
3. Split into blocks (8x8, typically), subtract 128 [to center around '0']
4. For each block
   a. Compute DCT coefficients
   b. Coarsely quantize
      - Many high frequency components will become zero
   c. Encode (with run length encoding and then Huffman coding for leftovers)

http://en.wikipedia.org/wiki/YCbCr
http://en.wikipedia.org/wiki/JPEG