Computational Photography and Image Manipulation
CMPT 469 / 985, Fall 2019

Week 3 cont’d
Short recap and an intro to JPEG compression

Slide credits

- Many slides thanks to James Tompkin along with his acknowledgements.

- Programming assignment 1 is released on the course website and CourSys
  - Deadline: October 8

- Our TA, Sicong Tang, has office hours on Mondays between 3-4pm at TASC8004

- **October 1**: Field trip to CS Research Day, link sent as news item on CourSys

- P2 papers – presentations on **October 3, Thursday**
  - Isola et al., Crisp Boundary Detection Using Pointwise Mutual Information, ECCV 2014
    - Presentation:
    - Discussion:
  - Raskar et al., Coded Exposure Photography: Motion Deblurring using Fluttered Shutter, SIGGRAPH 2006
    - Presentation:
    - Discussion:

Fourier Transform

•We want to understand the frequency ω of our signal. So, let's reparametrize the signal by ω instead of *x*:

For every ω from 0 to inf, ***F(ω)*** holds the amplitude *A* and phase ϕ of the corresponding sine $A\sin(\omega x + \phi)$

- How can *F* hold both?

$$F(\omega) = R(\omega) + iI(\omega)$$

$$A = \pm\sqrt{R(\omega)^2 + I(\omega)^2} \qquad \phi = \tan^{-1}\frac{I(\omega)}{R(\omega)}$$

We can always go back:

The Convolution Theorem

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

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

- The inverse Fourier transform of the product of two Fourier transforms is the convolution of the two inverse Fourier transforms

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

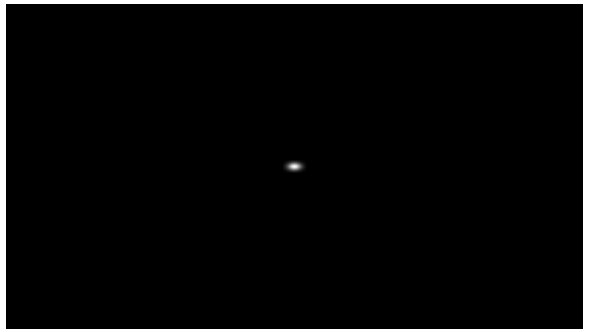
- **Convolution** in spatial domain is equivalent to **multiplication** in frequency domain!

2D convolution theorem example

$f(x,y)$

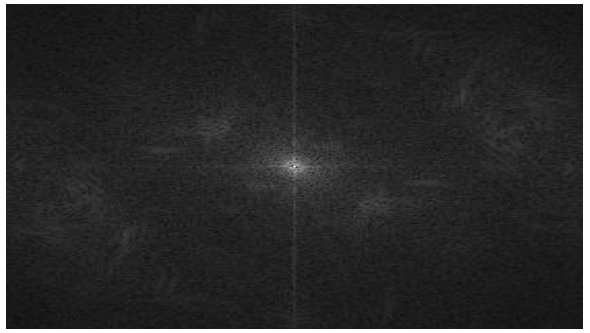
$*$

$h(x,y)$

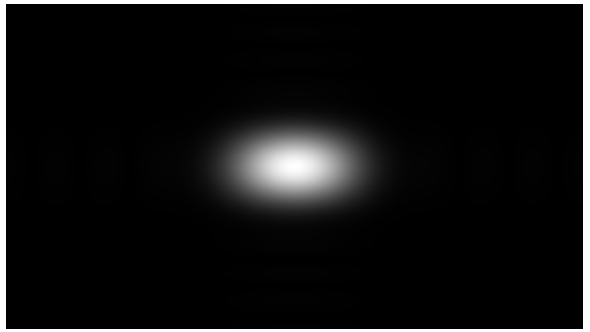


$\Downarrow$

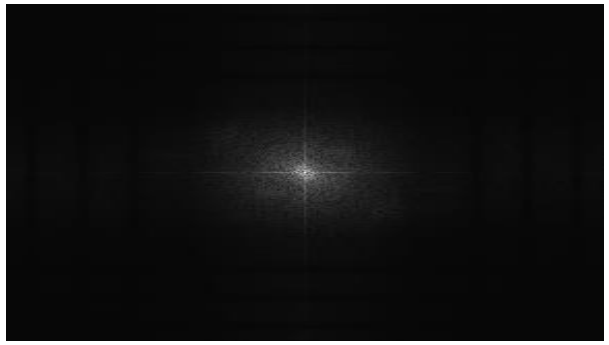
$g(x,y)$



$\asymp$



$\Downarrow$



$|F(s_x,s_y)|$

$|H(s_x,s_y)|$

$|G(s_x,s_y)|$

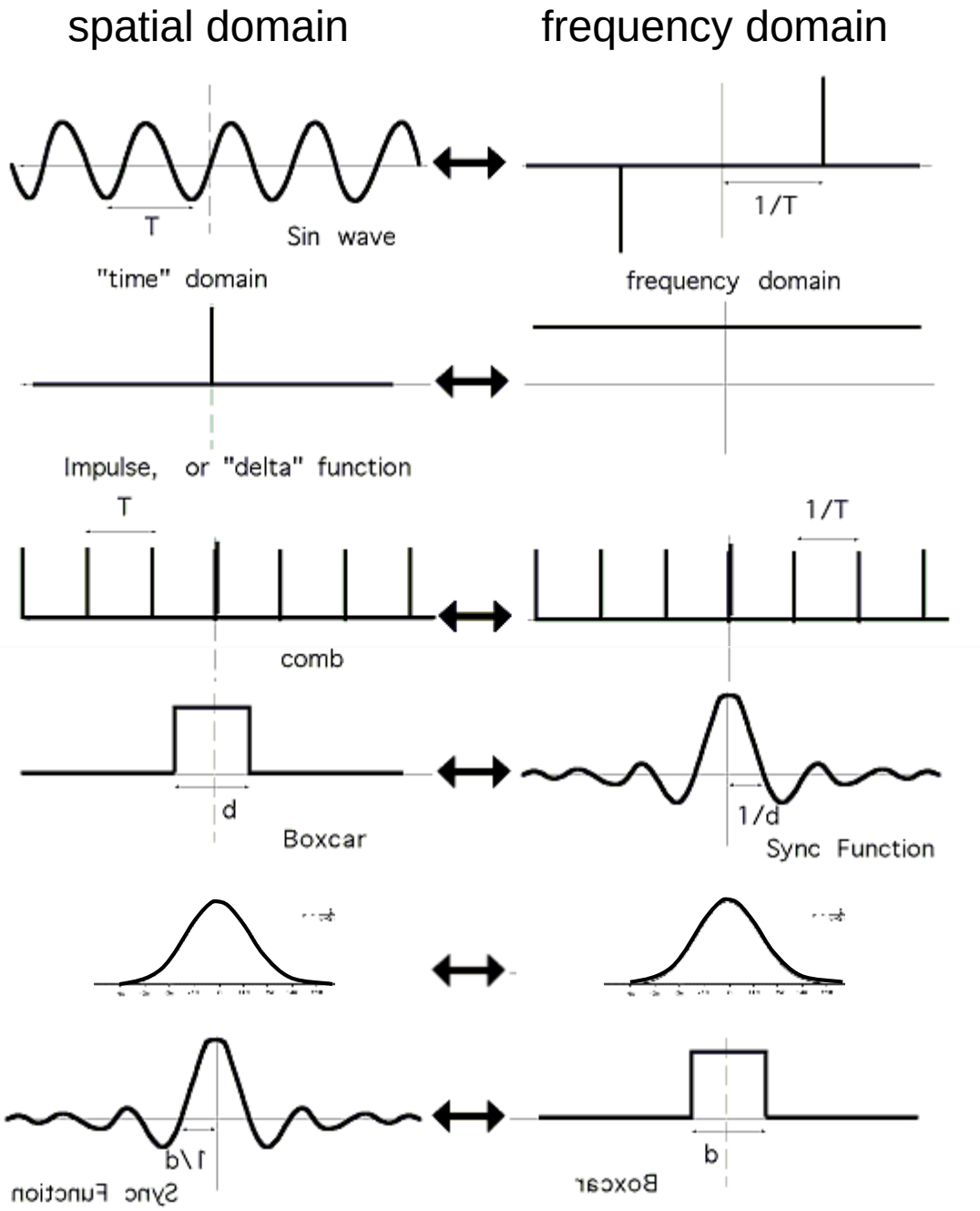
Low Pass filter

A low-pass filter attenuates high frequencies and retains low frequencies unchanged. The result in the spatial domain is equivalent to that of a smoothing filter; as the blocked high frequencies correspond to sharp intensity changes, *i.e.* to the fine-scale details and noise in the spatial domain image.

High Pass filter

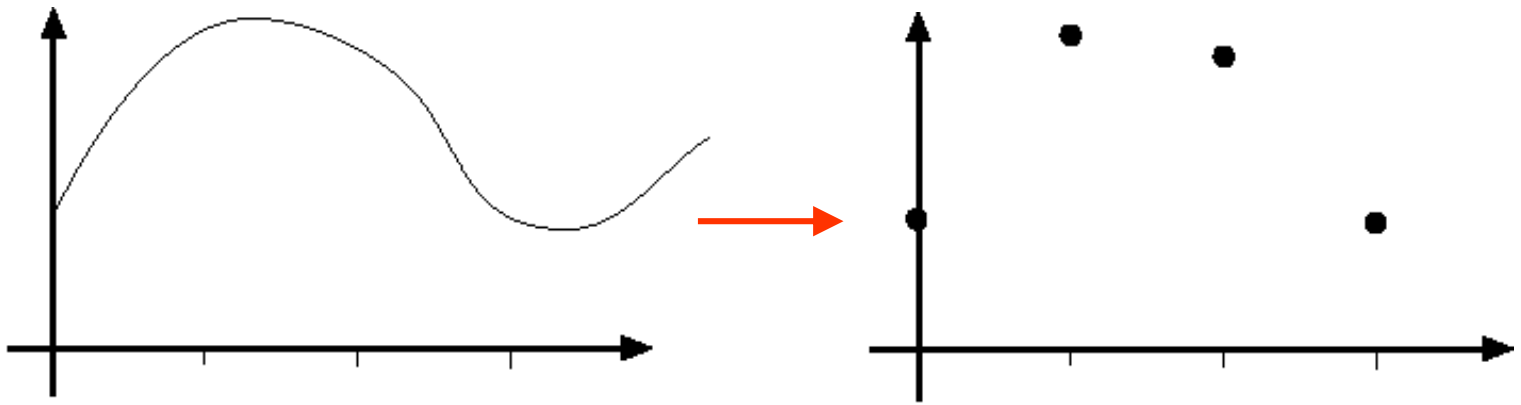
A highpass filter, on the other hand, yields edge enhancement or edge detection in the spatial domain, because edges contain many high frequencies. Areas of rather constant gray level consist of mainly low frequencies and are therefore suppressed.

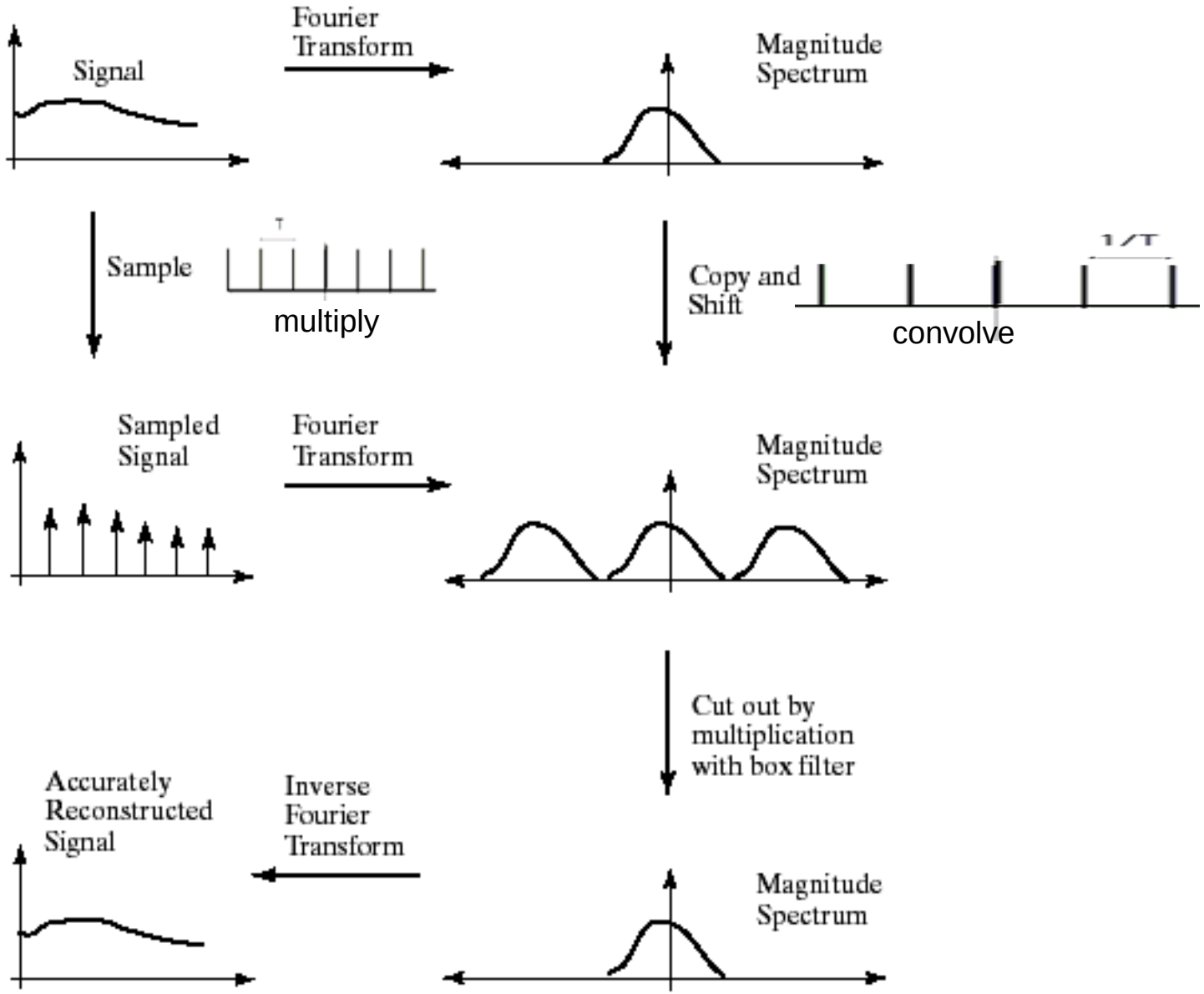
Fourier Transform of important functions

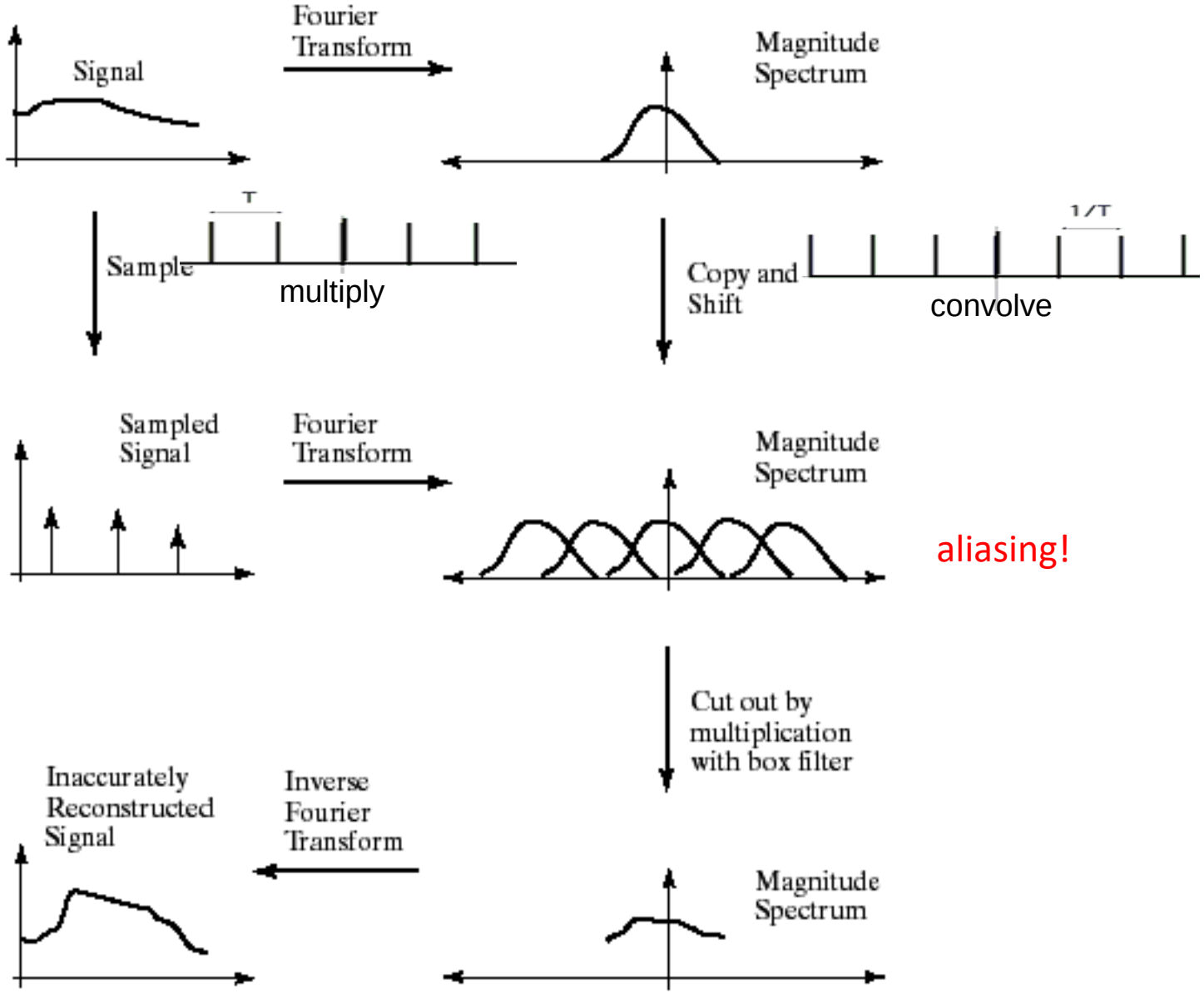


Sampling

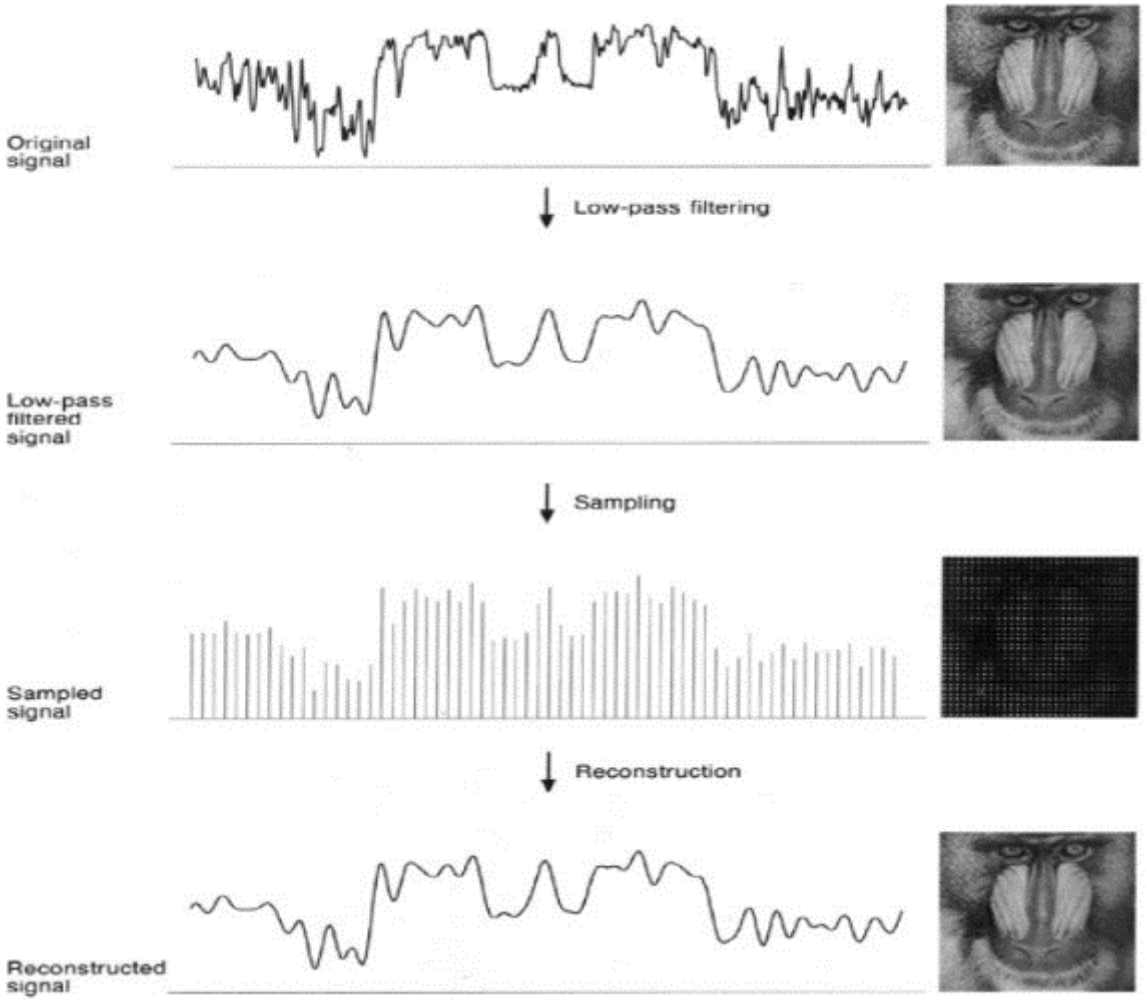
- Go from continuous world to discrete world, from function to vector
- Samples are typically measured on regular grid




Signal
Fourier Transform
Magnitude Spectrum
Sample
multiply
Copy and Shift
convolve
Sampled Signal
Fourier Transform
Magnitude Spectrum
Cut out by multiplication with box filter
Accurately Reconstructed Signal
Inverse Fourier Transform
Magnitude Spectrum

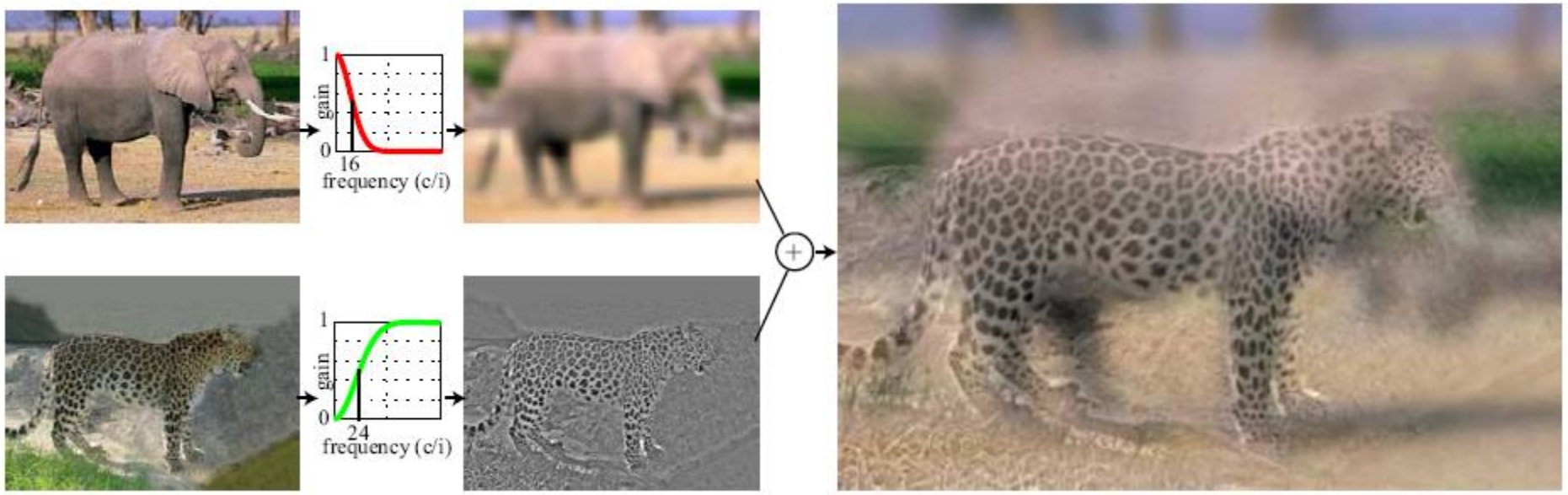

Signal
Fourier Transform
Magnitude Spectrum
T
Sample
multiply
1/T
Copy and Shift
convolve
Sampled Signal
Fourier Transform
Magnitude Spectrum
aliasing!
Cut out by multiplication with box filter
Inaccurately Reconstructed Signal
Inverse Fourier Transform
Magnitude Spectrum

Proper sampling



Slide from Dani Lischinski

Hybrid Images -- Homework



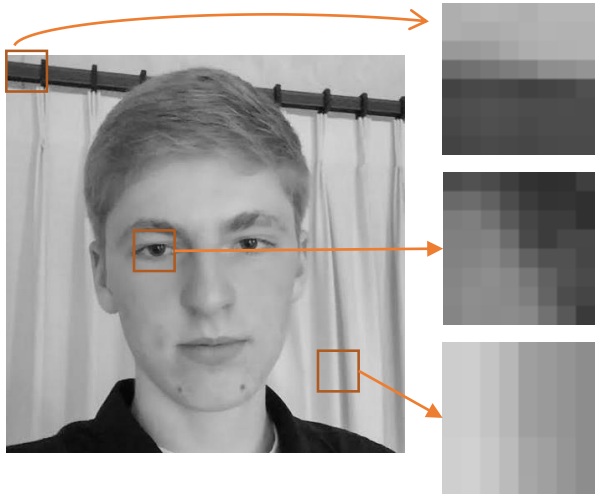
- A. Oliva, A. Torralba, P.G. Schyns, "Hybrid Images," SIGGRAPH 2006

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)

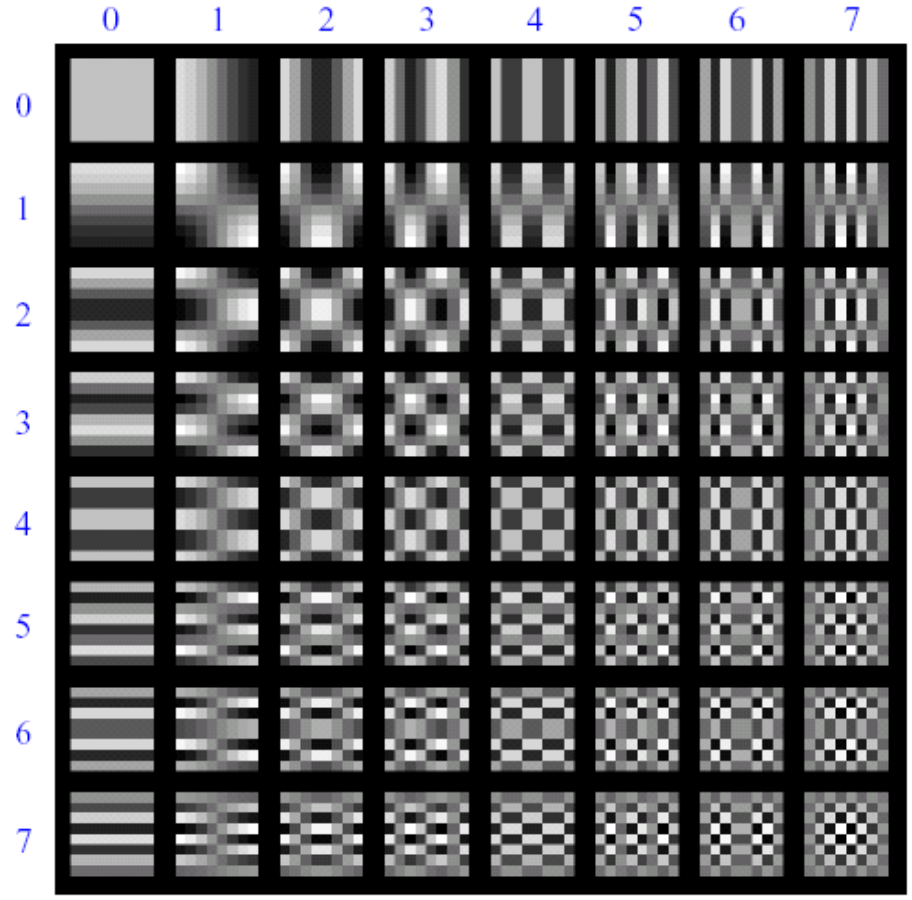
8x8 blocks



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

8x8 blocks



Block-based Discrete Cosine Transform (DCT)

Slides: Efros

Image compression using DCT

- Compute DCT filter responses in each 8x8 block

Filter responses

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

$u \longrightarrow$

- 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

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

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

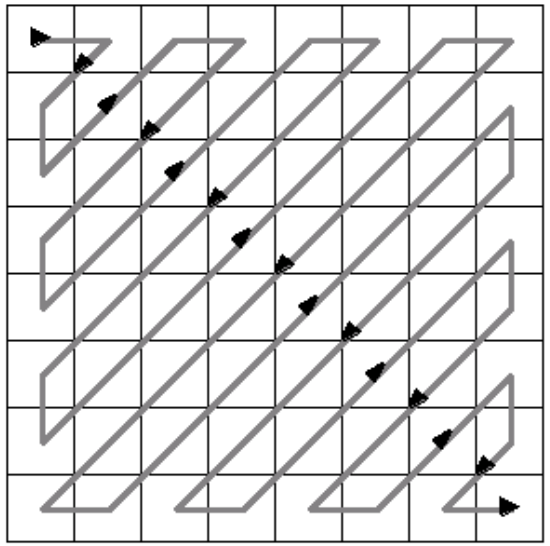
JPEG Encoding

- Entropy coding (Huffman-variant)

Quantized values

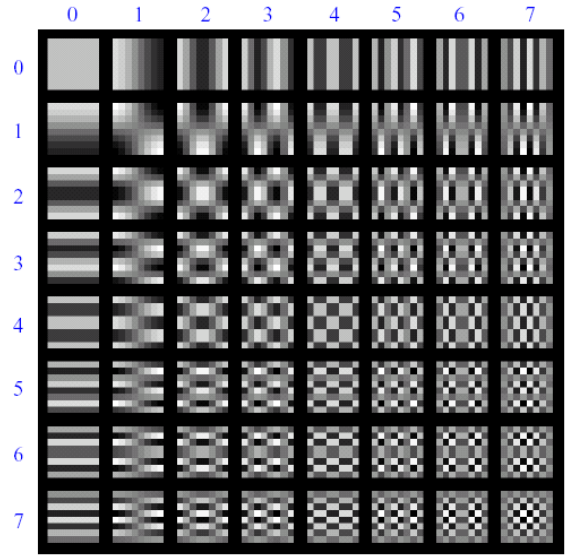
$$B = \begin{bmatrix} -26 & -3 & -6 & 2 & 2 & -1 & 0 & 0 \\ 0 & -2 & -4 & 1 & 1 & 0 & 0 & 0 \\ -3 & 1 & 5 & -1 & -1 & 0 & 0 & 0 \\ -3 & 1 & 2 & -1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Linearize *B* like this.



Helps compression:

- We throw away the high frequencies (‘0’).
- The zig zag pattern increases in frequency space, so long runs of zeros.



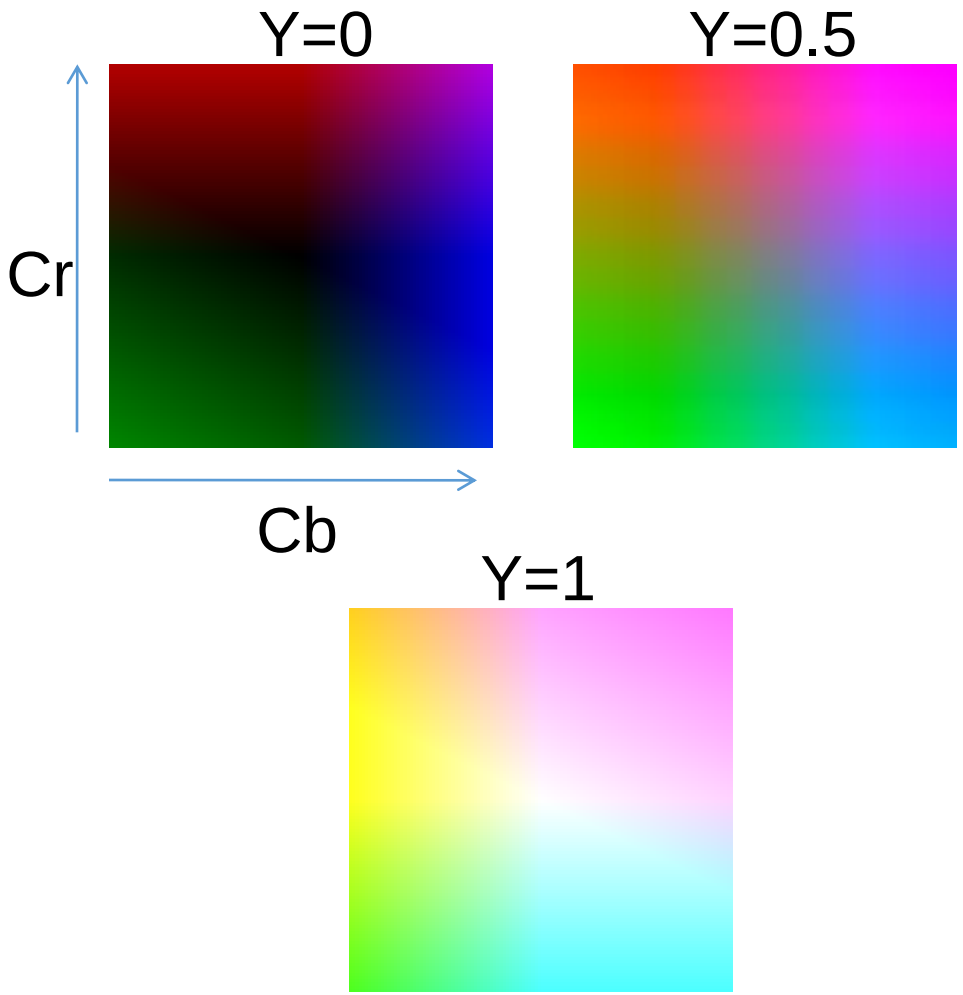
JPEG compression comparison

89k

12k

Color spaces: YCbCr

Fast to compute, good for compression, used by TV



Y
(Cb=0.5,Cr=0.5)

Cb
(Y=0.5,Cr=0.5)

Cr
(Y=0.5,Cb=05)

Most JPEG images & videos subsample chroma

PSP Comp 3
2x2 Chroma subsampling
285K

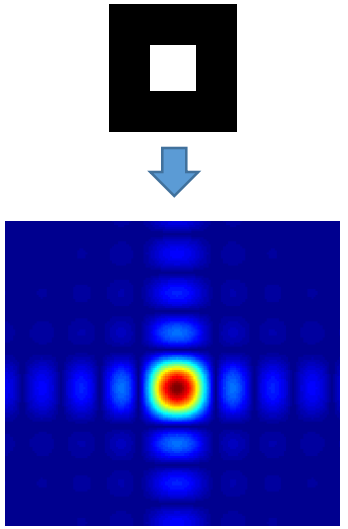
Original
1,261K lossless
968K PNG

JPEG Compression Summary

1. Convert image to YCrCb
2. Subsample color by factor of 2
   - People have bad resolution for color
3. Split into blocks (8x8, typically), subtract 128
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

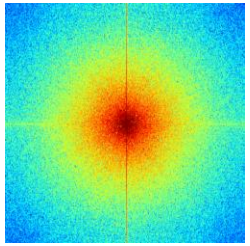
Things to Remember



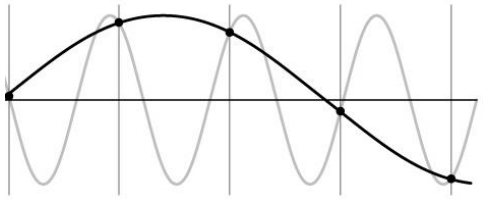
- Sometimes it makes sense to think of images and filtering in the frequency domain
  - Fourier analysis

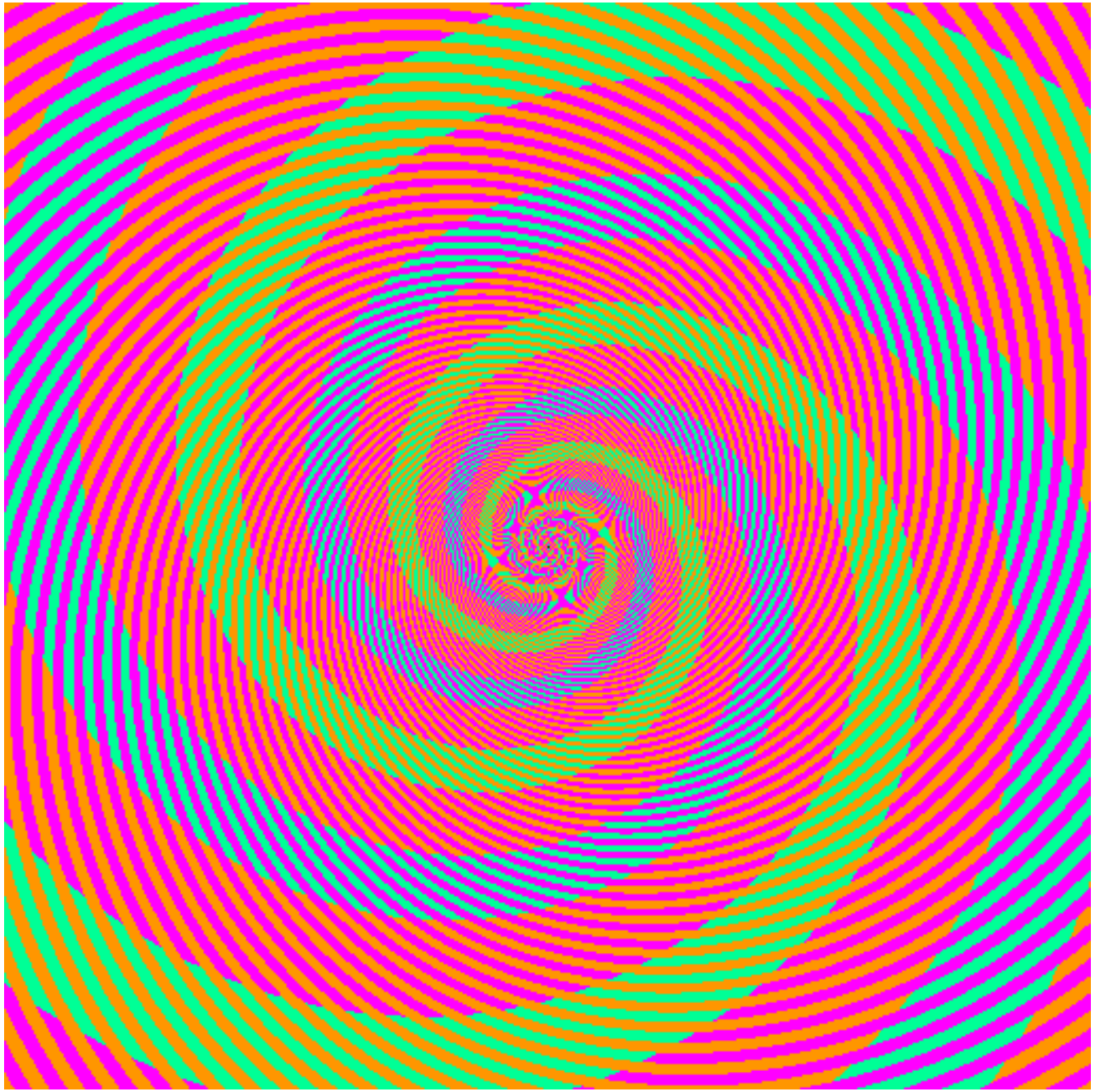
- Can be faster to filter using FFT for large images (N logN vs. N^2 for auto-correlation)

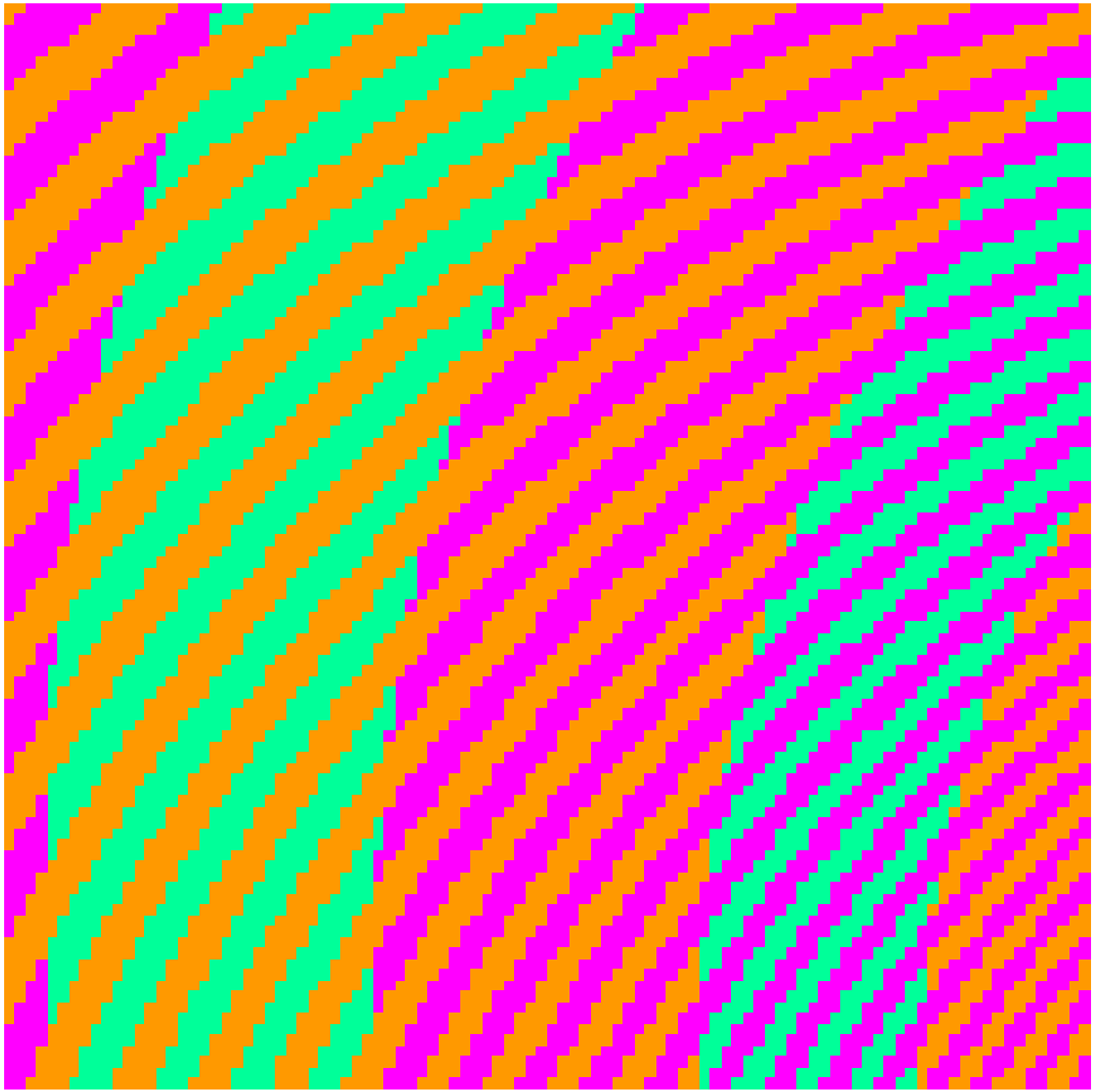
- Images are mostly smooth
  - Basis for compression

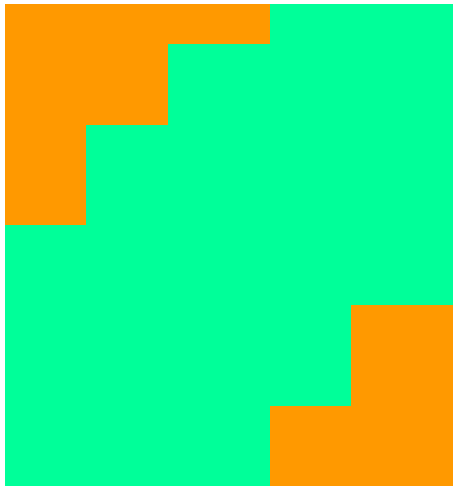


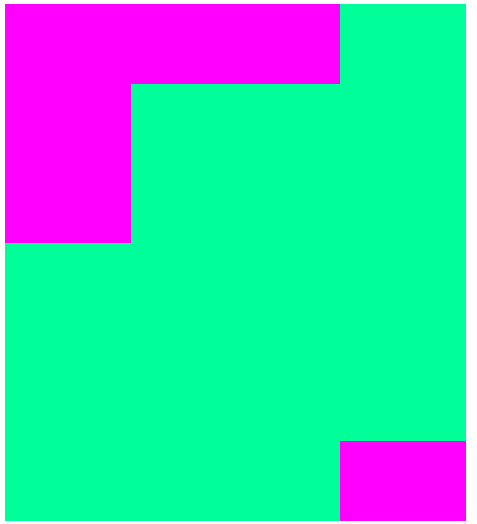
- Remember to low-pass before sampling











The blue and green colors are actually the same
http://blogs.discovermagazine.com/badastronomy/2009/06/24/the-blue-and-the-green/