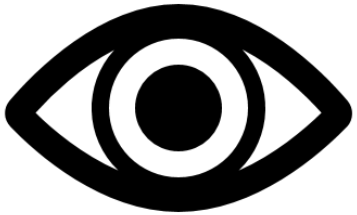


Advanced Image Processing

Thinking in Frequency

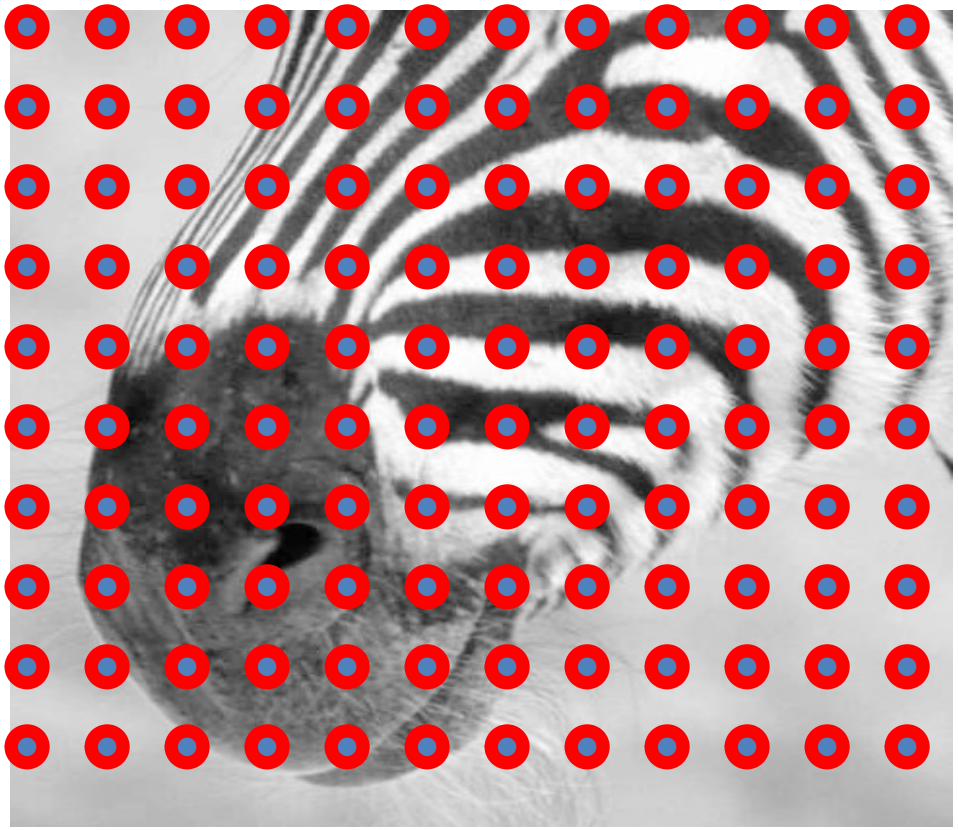
SYDE 671

Topics

- Spatial frequency
- Fourier transform and frequency domain
  - Frequency view of filtering
  - Hybrid images

Slide: Hoiem

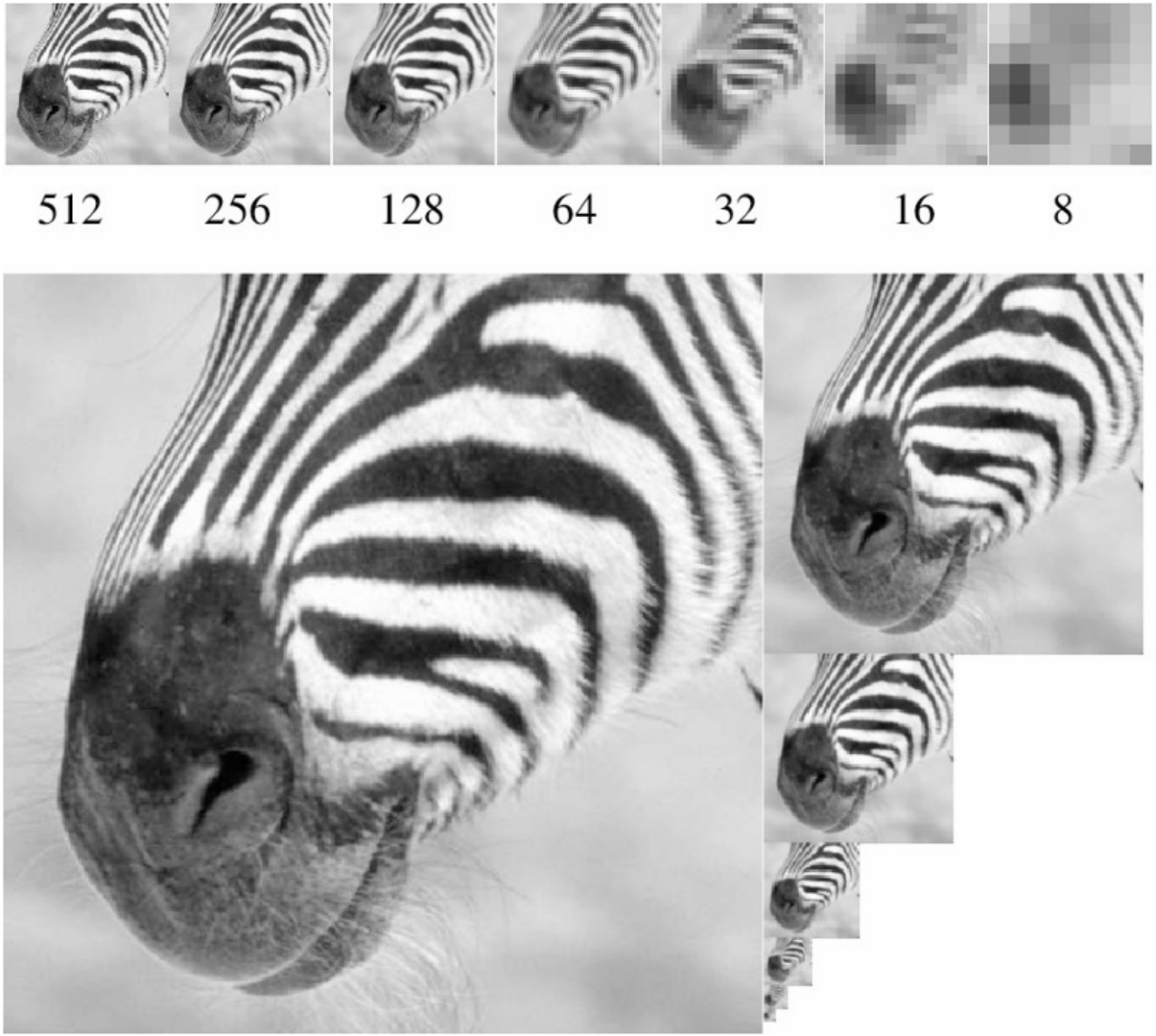
Subsampling by a factor of 2



Throw away every other row and column to create a 1/2 size image

Why does a lower resolution image still make sense to us? What information do we lose?

Slide: Hoiem



A ‘bar’ in the big images is a hair on the zebra’s nose; in smaller images, a stripe; in the smallest, the animal’s nose

Figure from David Forsyth

Algorithm for downsampling by factor of 2

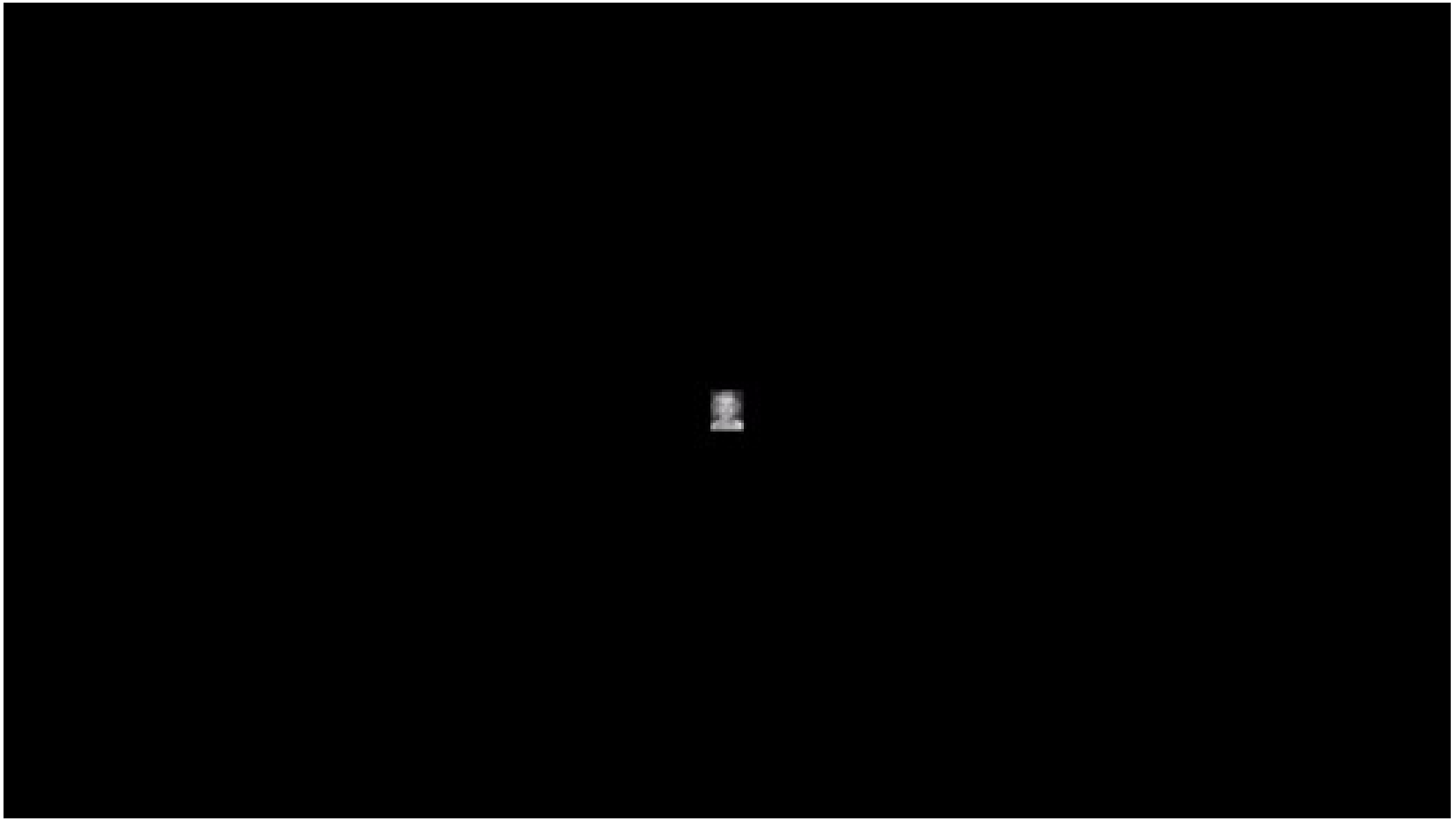
1. Start with image of w x h
2. Sample every other pixel
   - im_small = image[::2:, ::2]
3. Repeat until im_small is 1 pixel large.

Numpy syntax:
::2 -> start at 0, end at 'end', increase every 2, until the end.
e.g., 0,2,4,6,...,w

(if w is not even, then this goes to w-1)

What can go wrong??

Einstein - Marilyn Monroe



https://www.youtube.com/watch?v=gfvMU36fgKw

Salvador Dali, 1976

Why do we get different, distance-dependent interpretations of hybrid images?

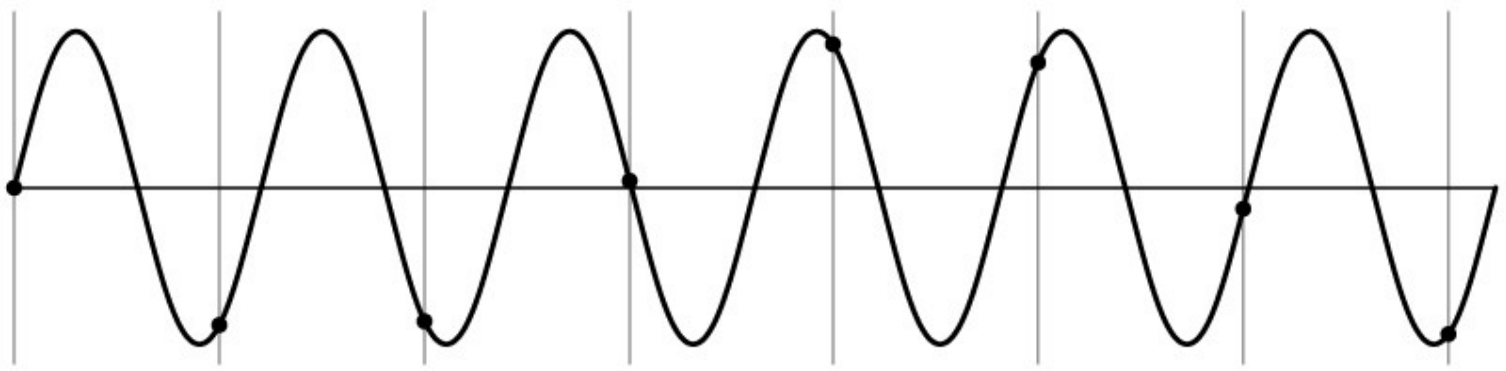
A. Oliva, A. Torralba, P.G. Schyns, "Hybrid Images," SIGGRAPH 2006

?

Slide: Hoiem

Aliasing problem

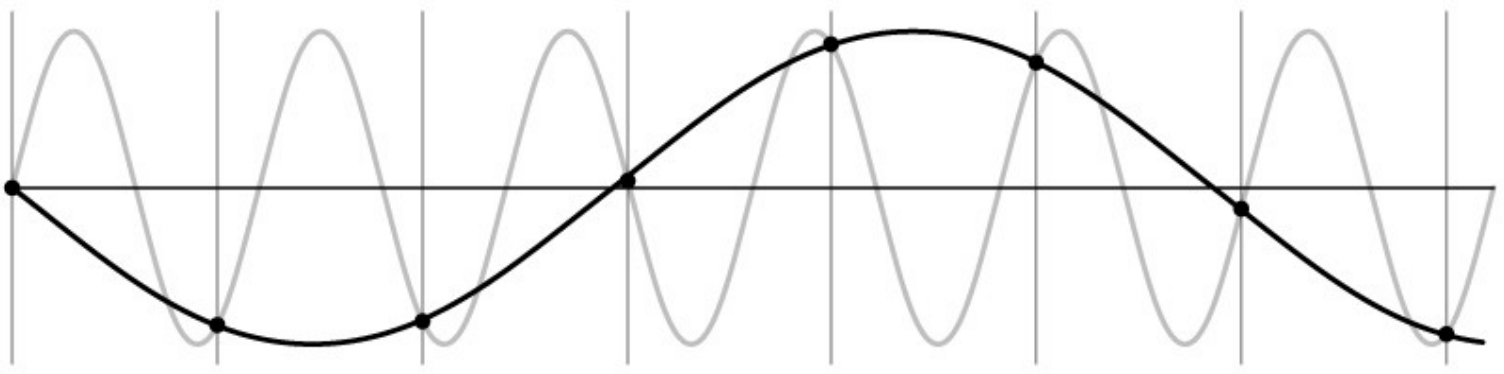
- 1D example (sinewave):



Source: S. Marschner

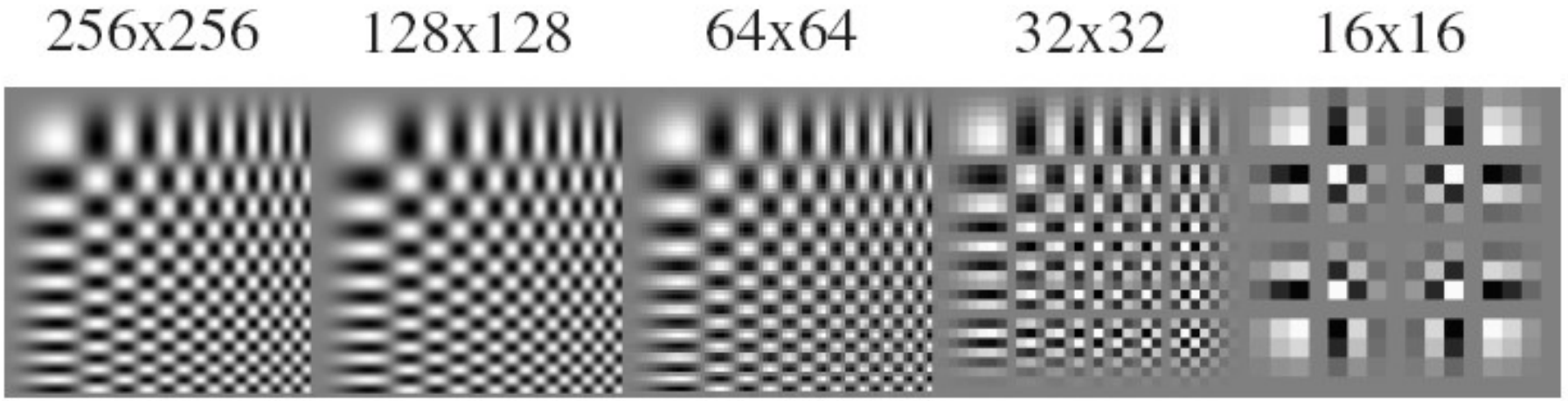
Aliasing problem

- 1D example (sinewave):



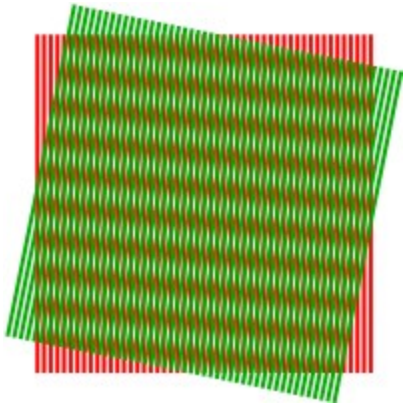
Source: S. Marschner

Sampling and aliasing



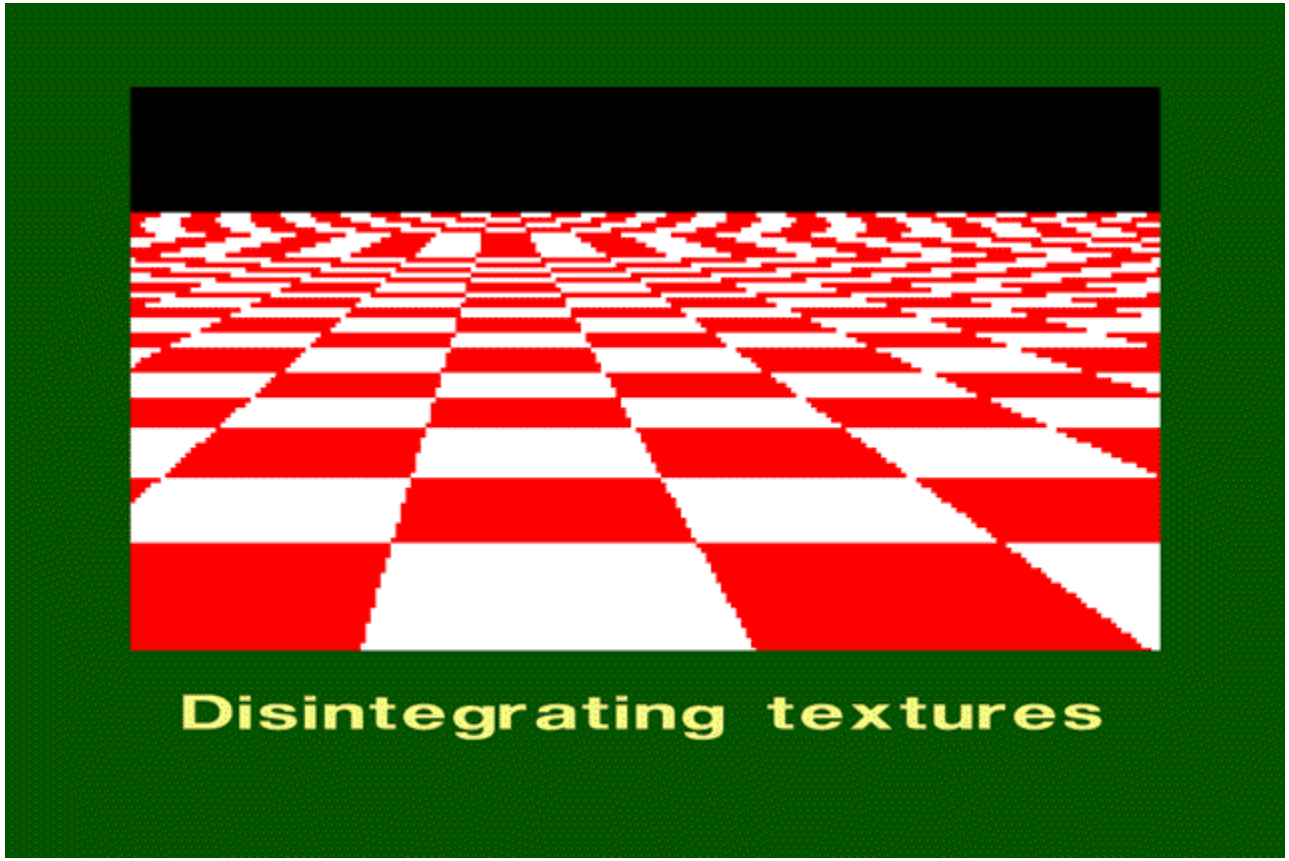
Aliasing problem

- Sub-sampling may be dangerous….
- Characteristic errors may appear:
  - “car wheels rolling the wrong way in movies”
  - “checkerboards disintegrate in ray tracing”
  - “striped shirts look funny on color television”
    - Moiré patterns



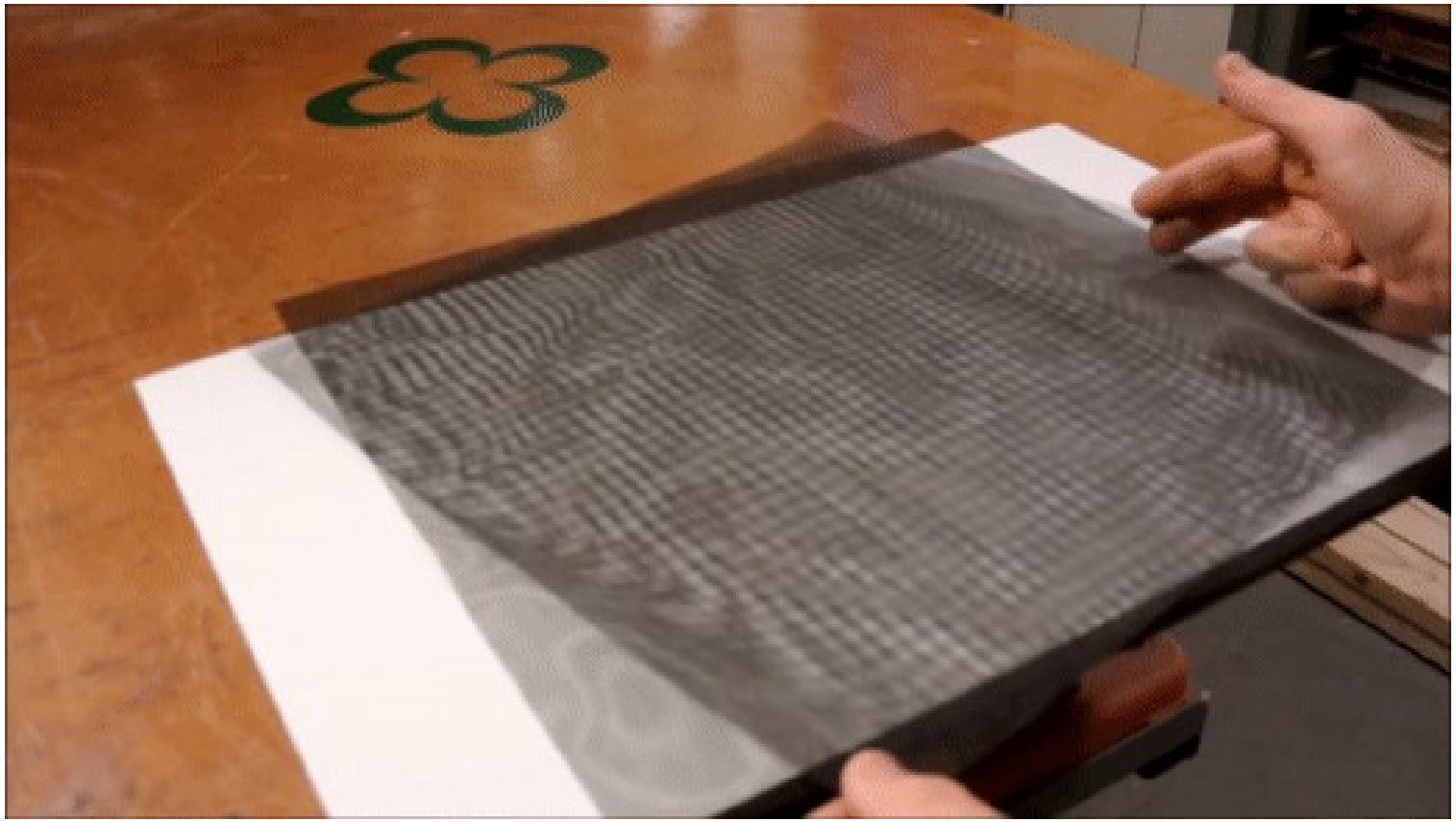
Source: D. Forsyth; Wikipedia

Spatial sampling: Aliasing in graphics



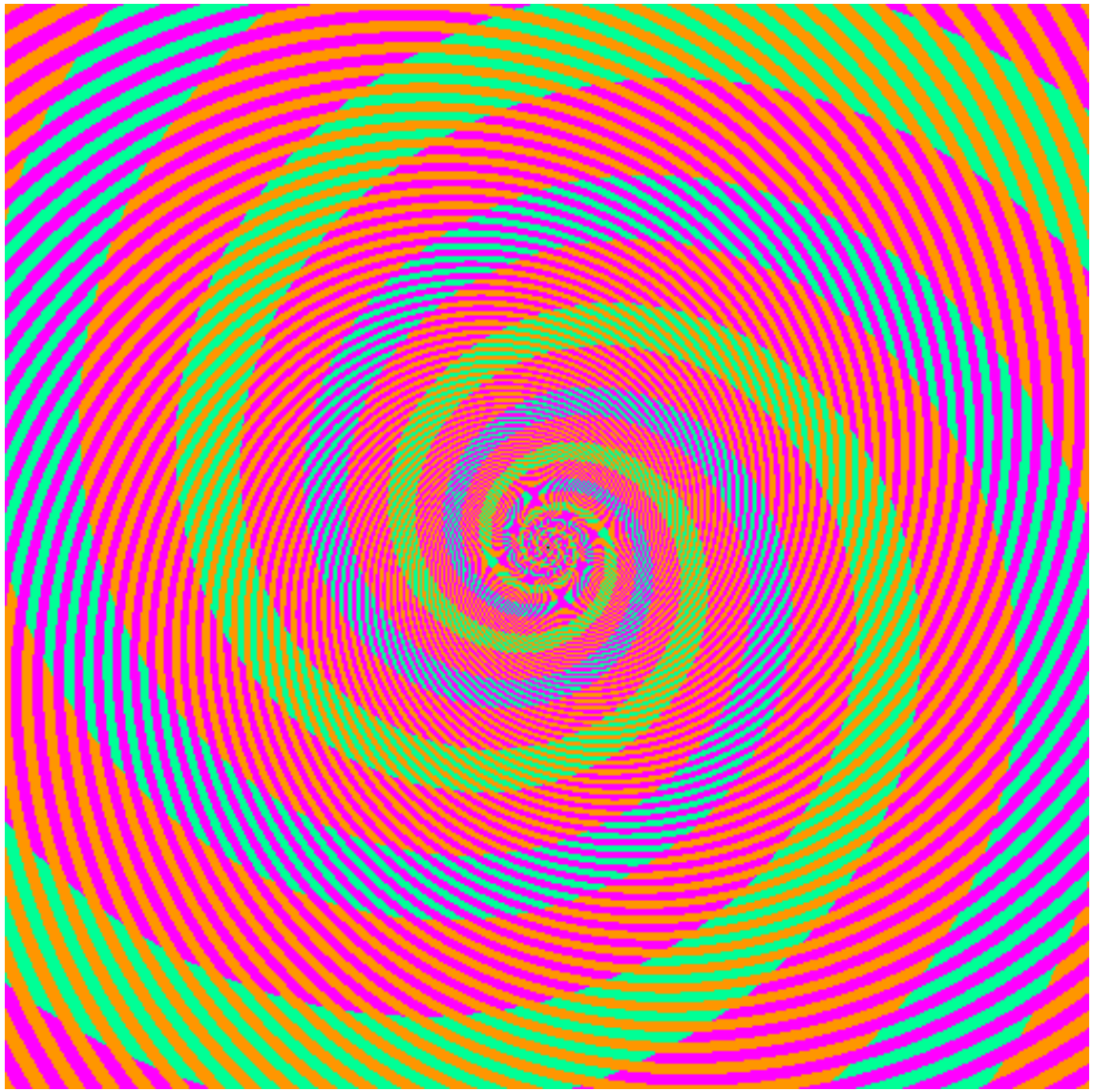
Source: A. Efros

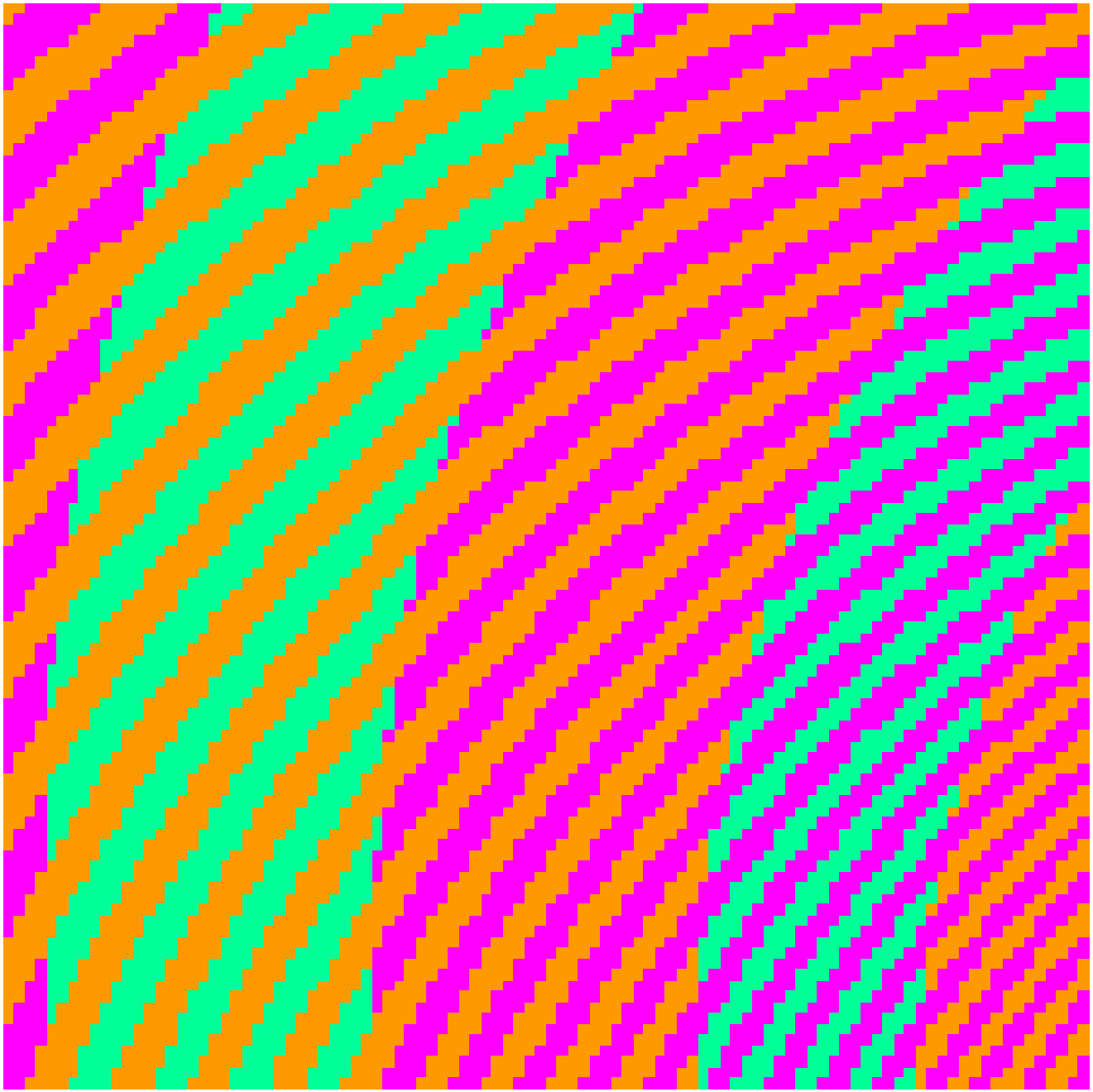
Spatial sampling: Aliasing and Moiré patterns



Youtube: pocket83²

Spatial sampling: Brain color segmentation malfunction





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

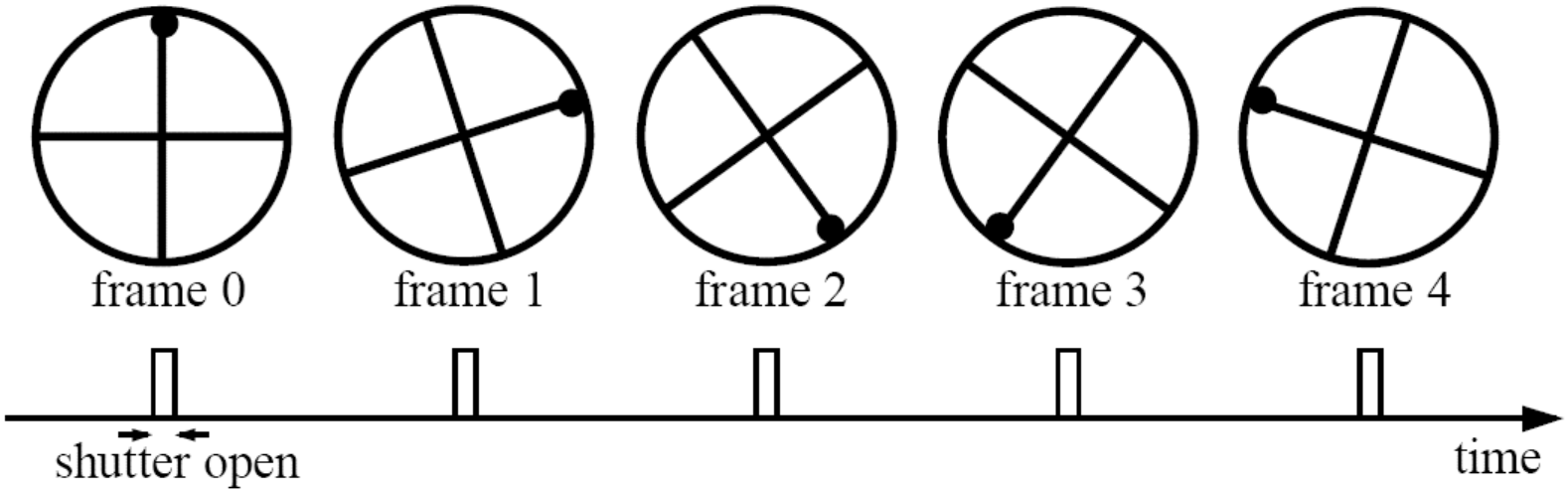
Videos

[YouTube; JoinBuzzirk]

Temporal sampling: Aliasing in video

Imagine a spoked wheel moving to the right (rotating clockwise). Mark wheel with dot so we can see what's happening.

If camera shutter is only open for a fraction of a frame time (frame time = 1/30 sec. for video, 1/24 sec. for film):



Without dot, wheel appears to be rotating slowly backwards! (counterclockwise)

Slide by Steve Seitz

Videos

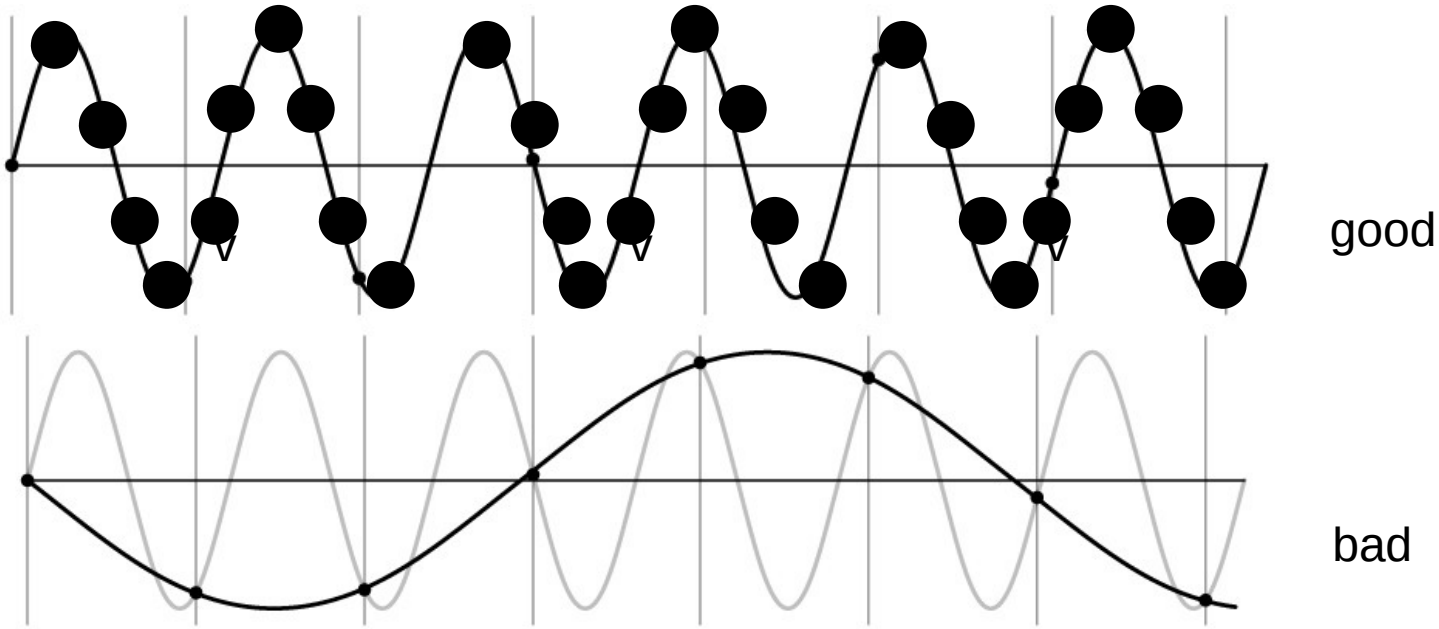
[YouTube; JoinBuzzirk]

Videos

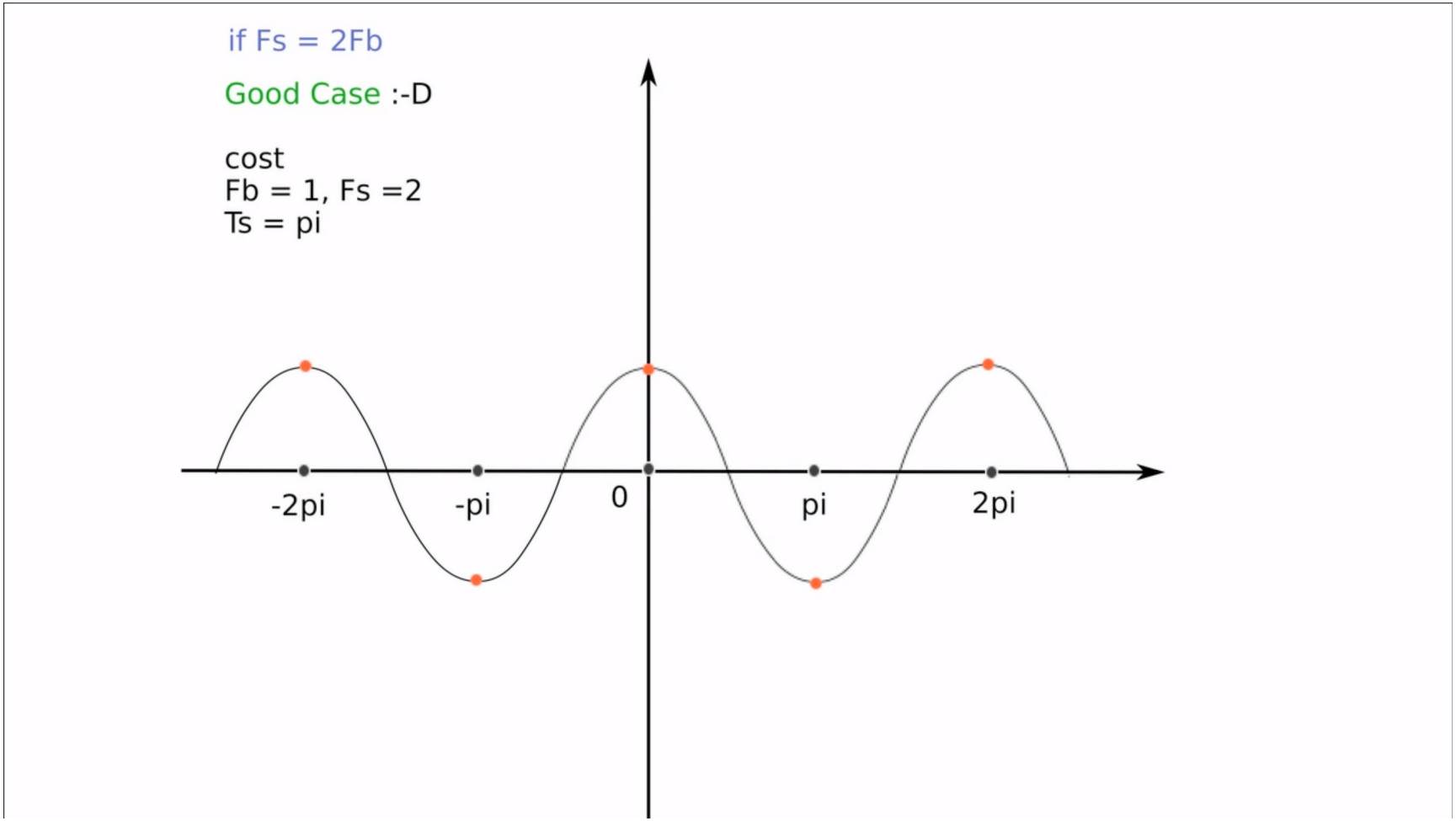
 [YouTube: phrancque]

Nyquist-Shannon Sampling Theorem

- When sampling a signal at discrete intervals, the sampling frequency must be $2 \; f_{max}$
- f_{max} = max frequency of the input signal
- This will allows to reconstruct the original perfectly from the sampled version

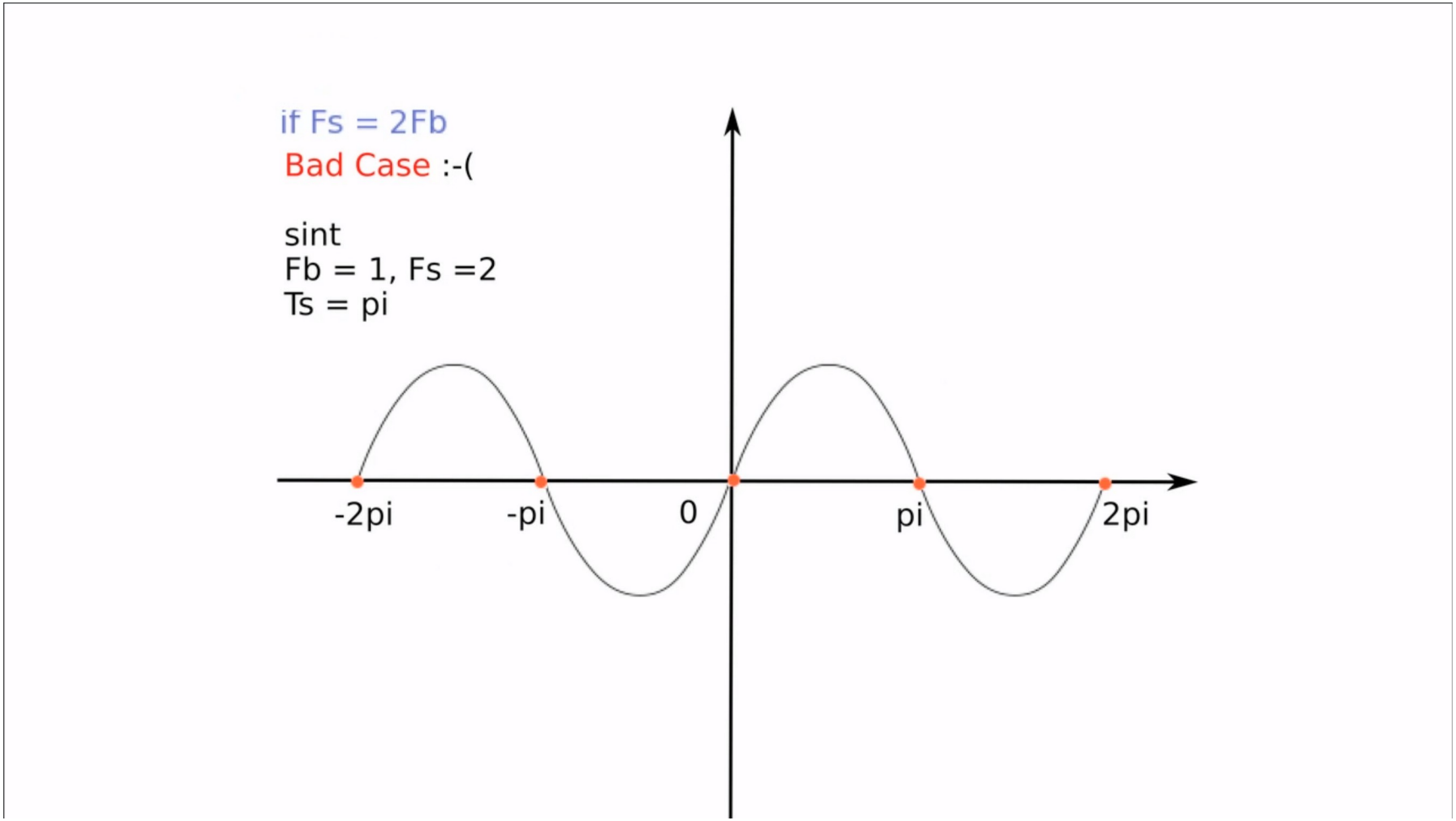


Nyquist-Shannon Sampling Theorem



Youtube: Carren Wang

Nyquist-Shannon Sampling Theorem



Youtube: Carren Wang

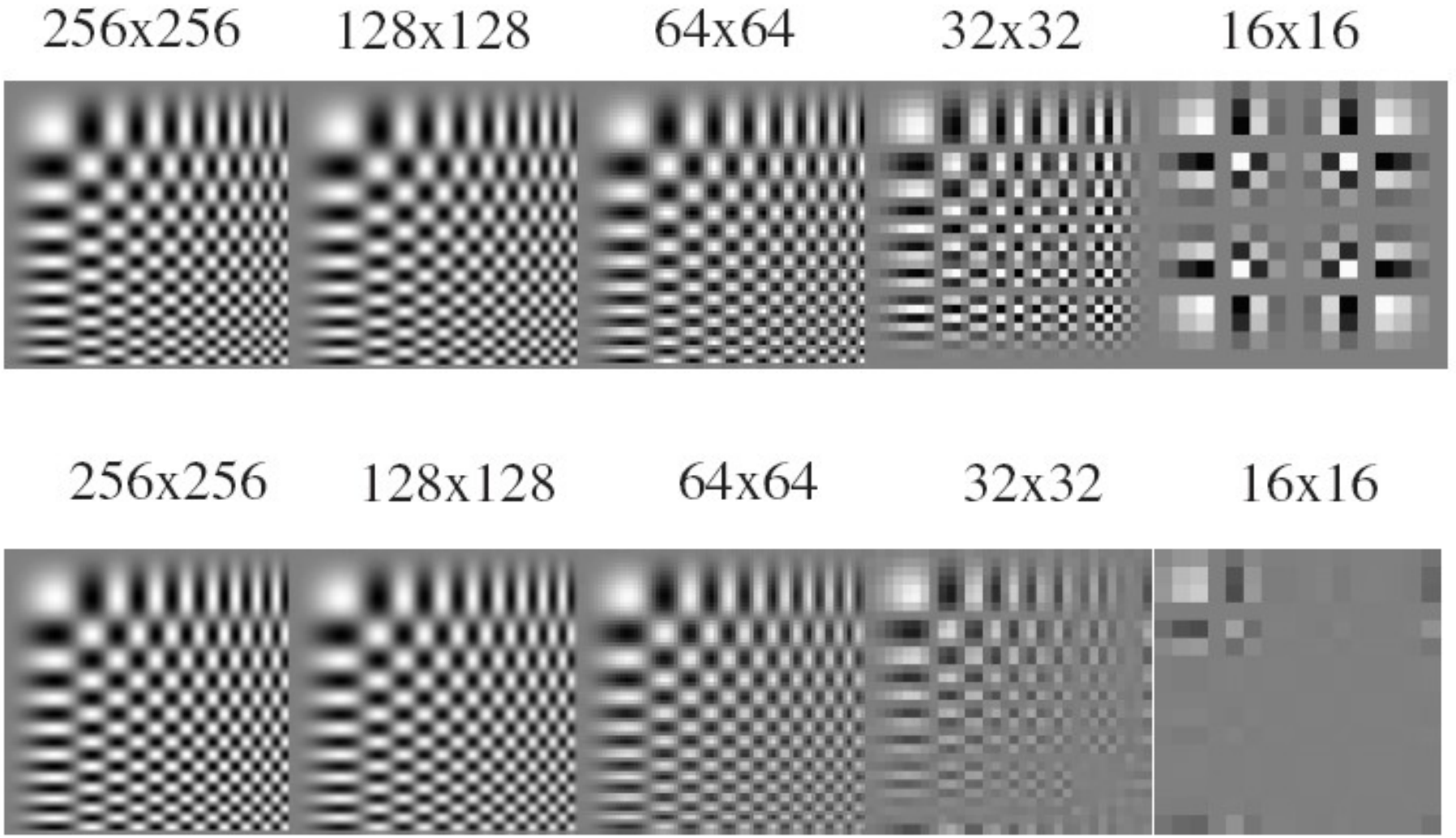
Anti-aliasing

Solutions:

- Sample more often (better sensors)

- Get rid of all frequencies that are greater than half the new sampling frequency
  - Will lose information
  - But it's better than aliasing
  - Apply a smoothing (*low pass*) filter

Anti-aliasing



Forsyth and Ponce 2002

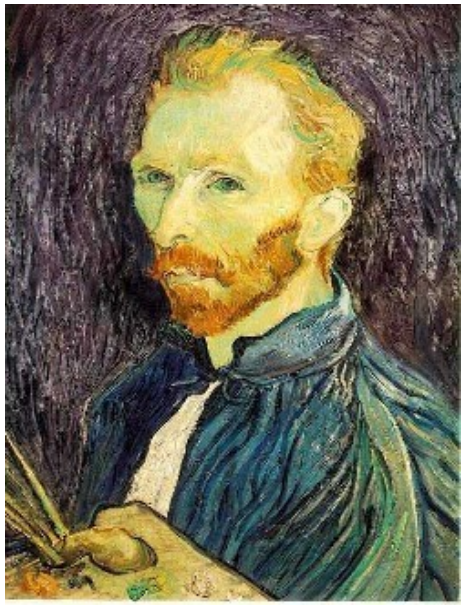
Algorithm for downsampling by factor of 2

1. Start with image(h, w)
2. Apply low-pass filter

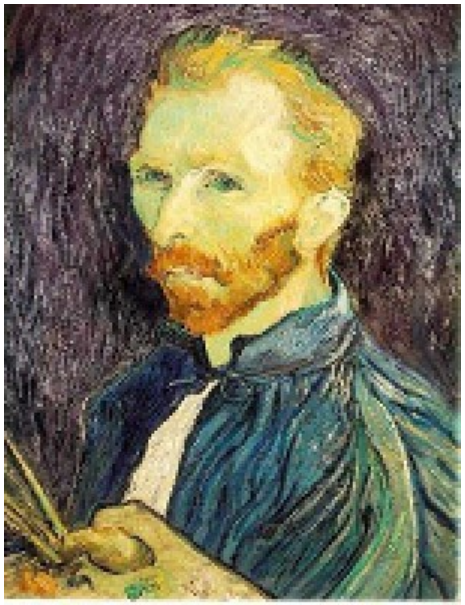
   im_blur = imfilter(image, fspecial(‘gaussian’, 7, 1))
3. Sample every other pixel

   im_small = im_blur(1:2:end, 1:2:end);

Subsampling without filtering



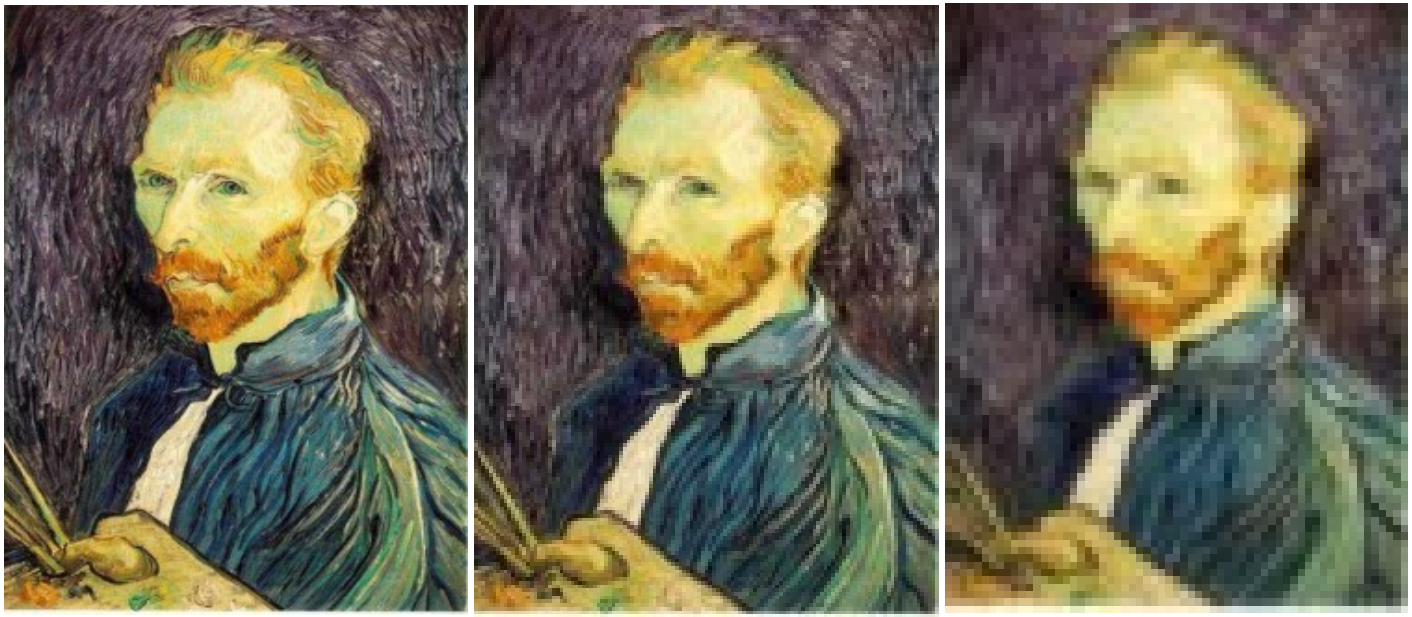
1/2



1/4 (2x subsample)

1/8 (4x subsample)

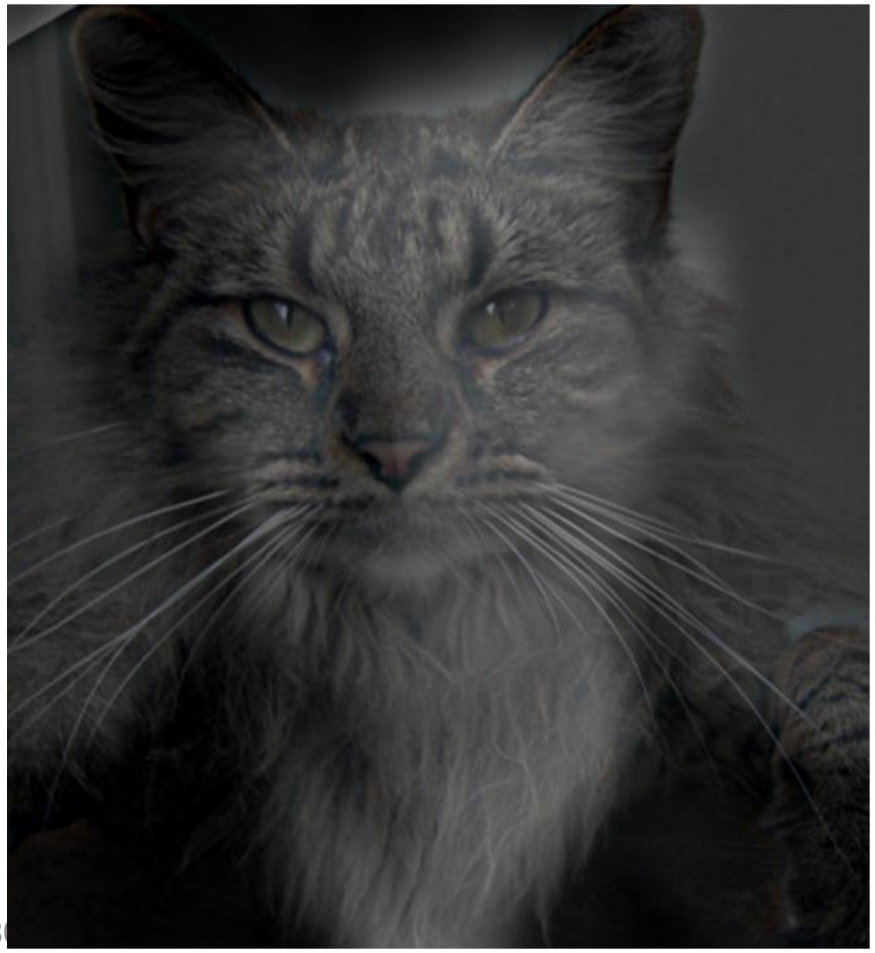
Subsampling with Gaussian pre-filtering



Gaussian 1/2 G 1/4 G 1/8

Gaussian Pyramid [Burt and Adelson, 1983]

Why do we get different, distance-dependent interpretations of hybrid images?

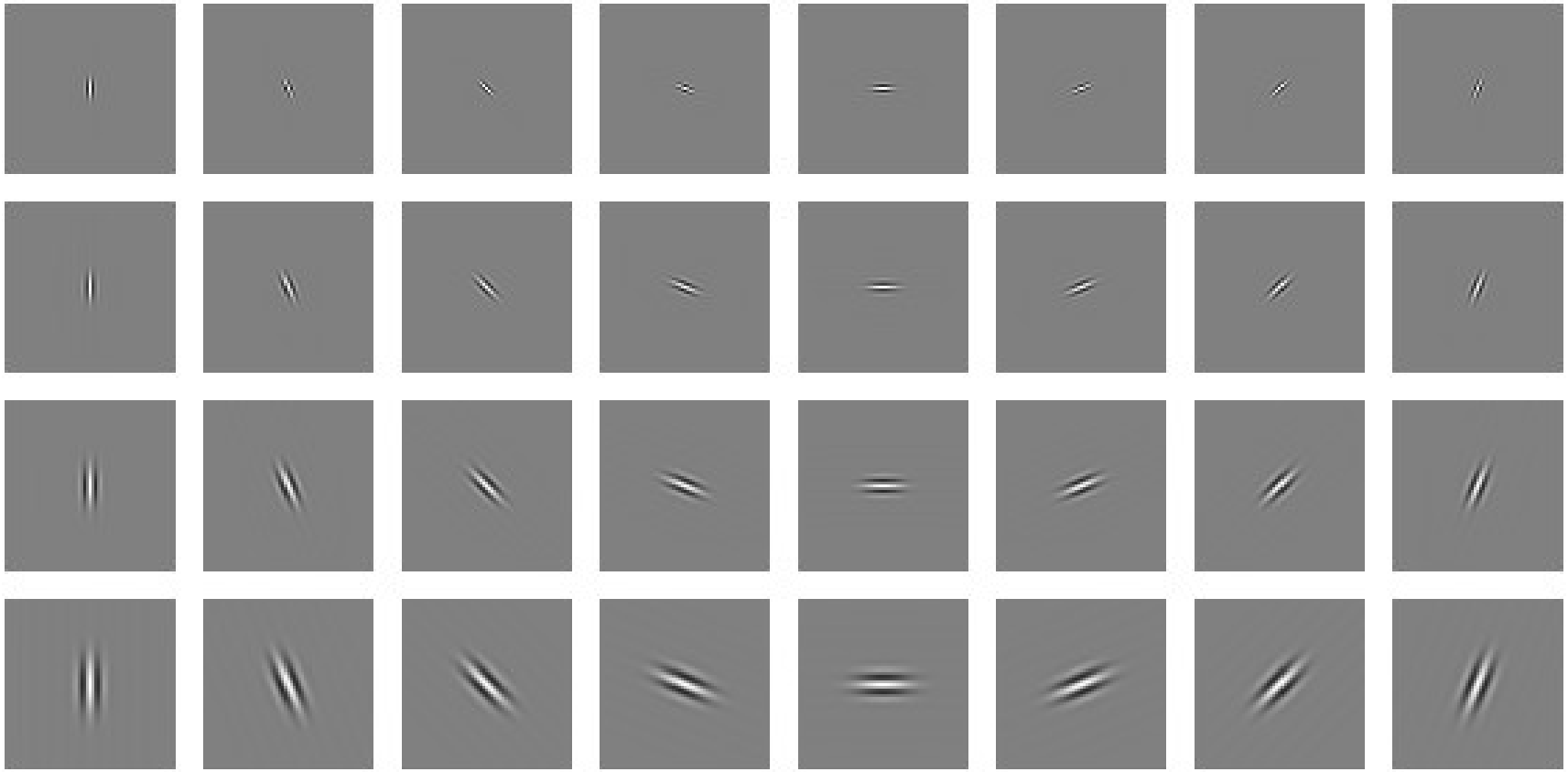


?

Hays

Clues from Human Perception

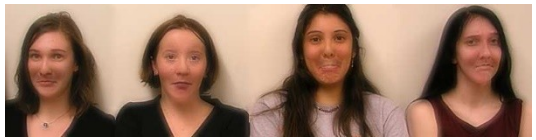
- Early processing in humans filters for orientations and scales of frequency.



Early Visual Processing: Multi-scale edge and blob filters

Application: Hybrid Images

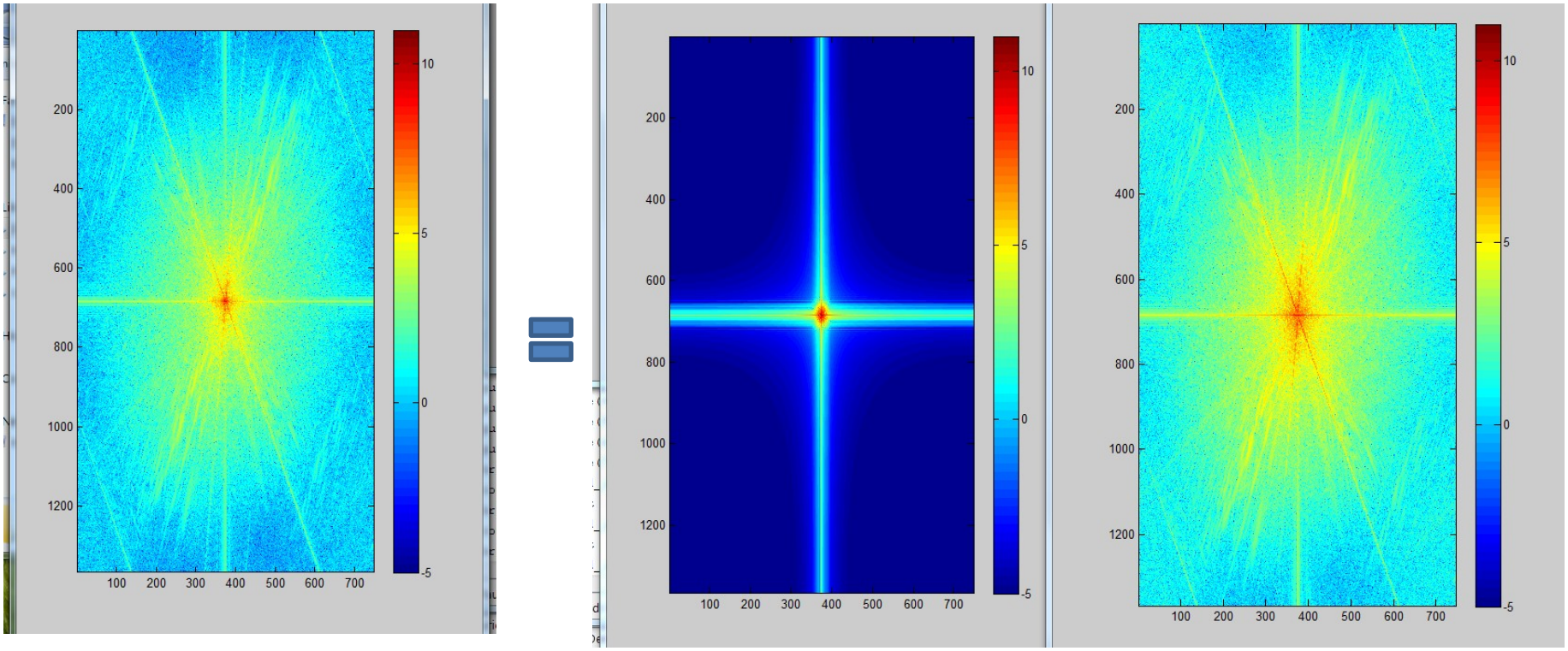
When we see an image from far away, we are effectively subsampling it!



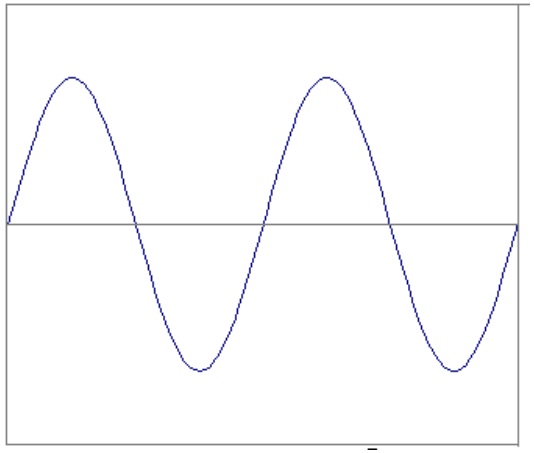
A. Oliva, A. Torralba, P.G. Schyns, SIGGRAPH 2006

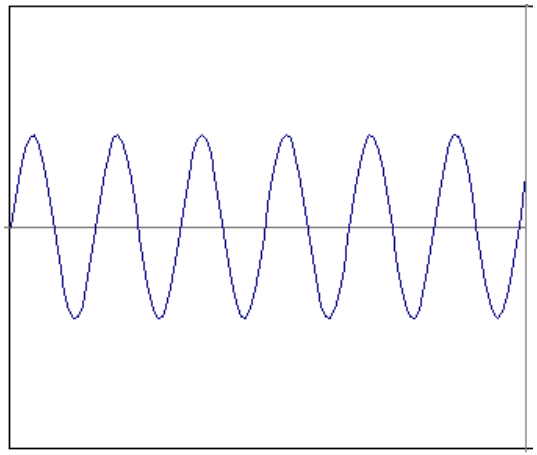
Hybrid Image in FFT

Hybrid Image = Low-passed Image + High-passed Image



I understand frequency as in waves…





…but how does this relate to the complex signals we see in natural images?

…to image frequency?

Fourier series

A bold idea (1807):

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

Our building block:

$$\sum_{n=1}^{\infty} \left(a_n \cos\left(nt\right) + b_n \sin\left(nt\right) \right)$$

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

Jean Baptiste Joseph Fourier (1768-1830)

Jean Baptiste Joseph Fourier (1768-1830)

A bold idea (1807):

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

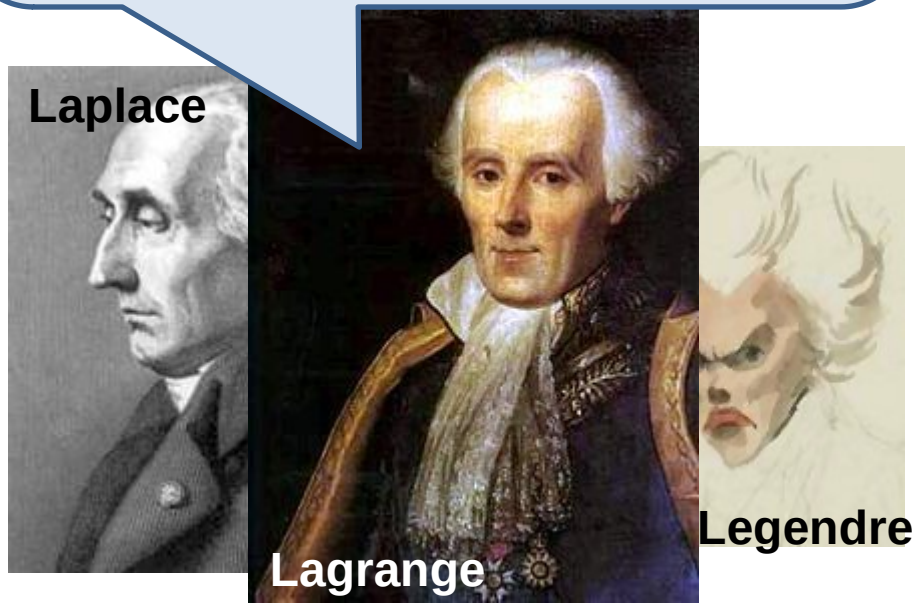
Don't believe it?

- Neither did Lagrange, Laplace, Poisson and other big wigs
- Not translated into English until 1878!

But it's (mostly) true!

- Called Fourier Series
- *Applies to periodic signals*

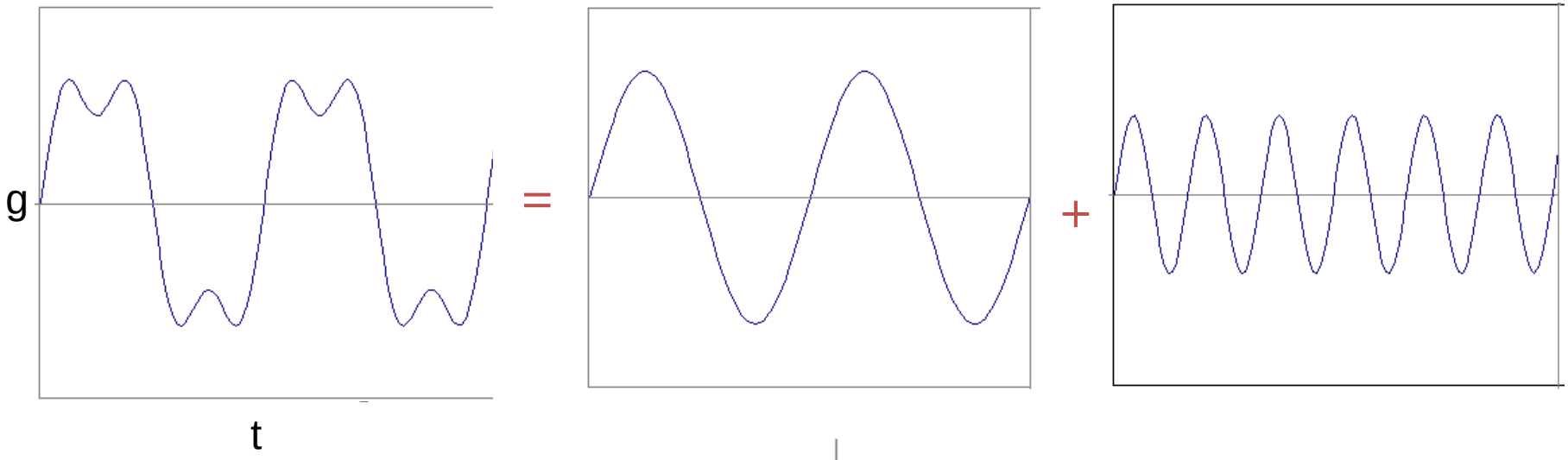
…the manner in which the author arrives at these equations is not exempt of difficulties and…his analysis to integrate them still leaves something to be desired on the score of generality and even rigour.

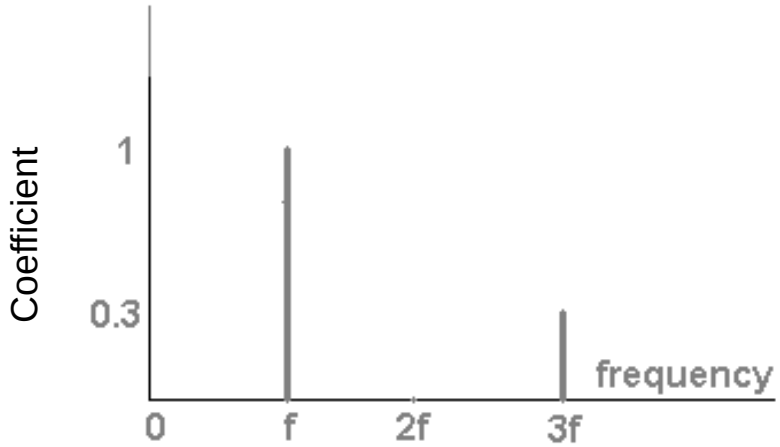


Reviewer #2

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

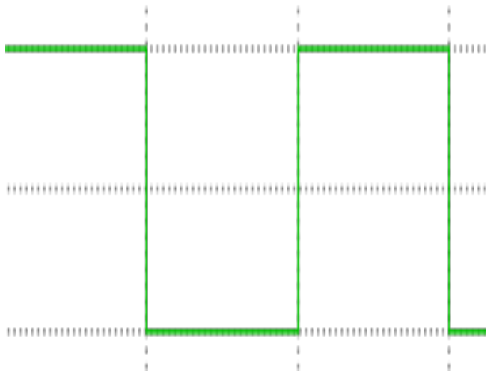
$$g(t) = (1)\sin(2\pi f\, t) + (1/3)\sin(2\pi(3f)\, t)$$



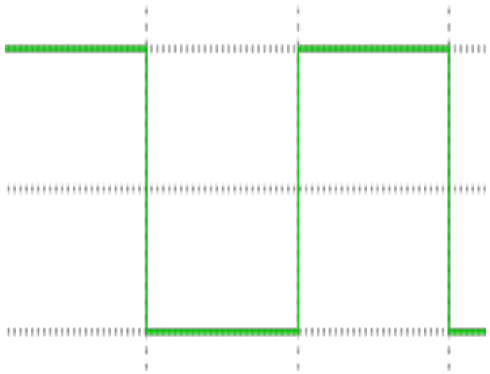


Slides: Efros

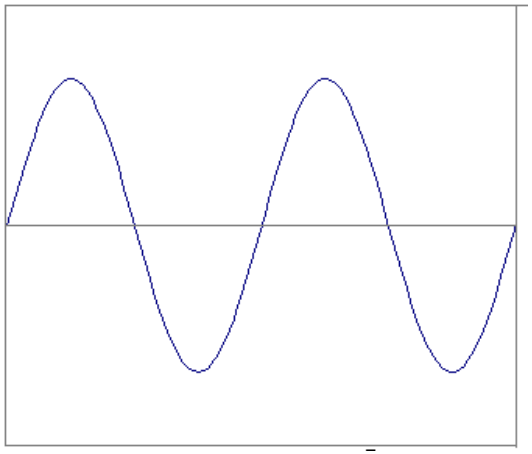
Square wave spectra



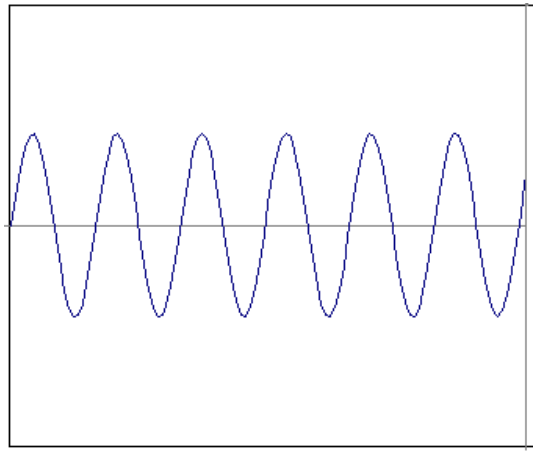
Square wave spectra



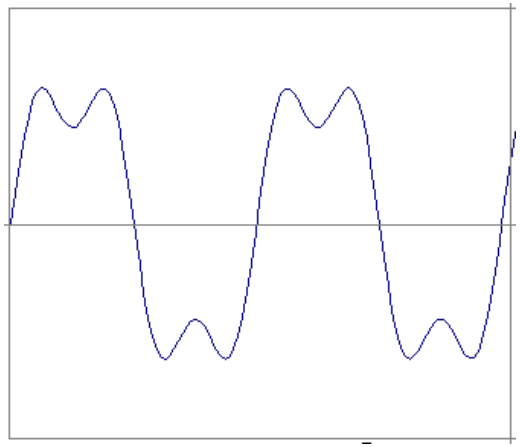
=



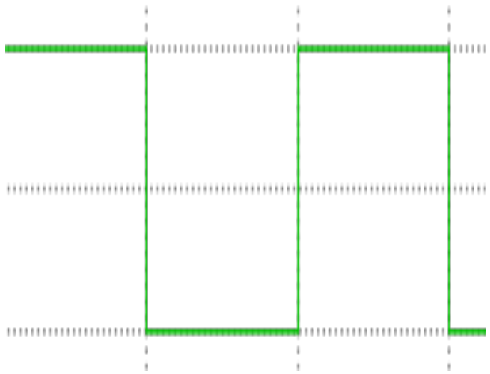
+



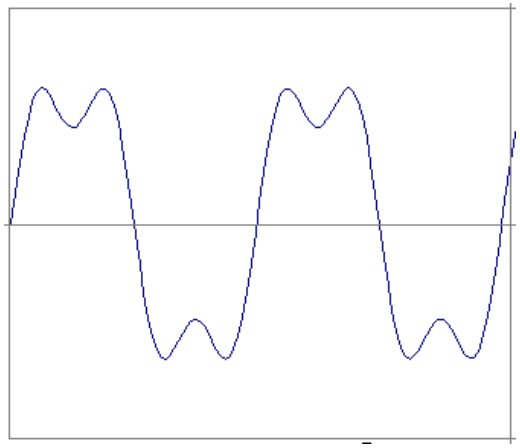
=



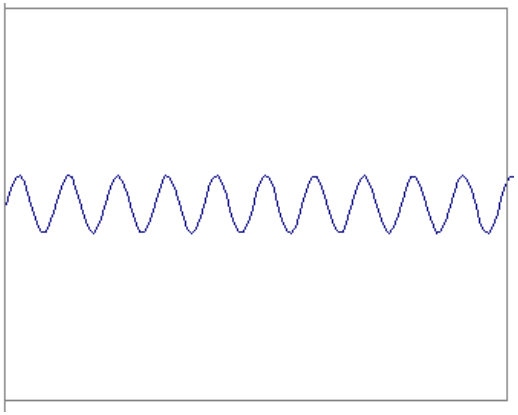
Square wave spectra



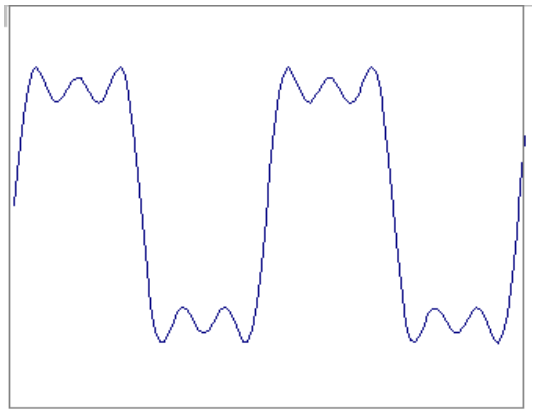
=



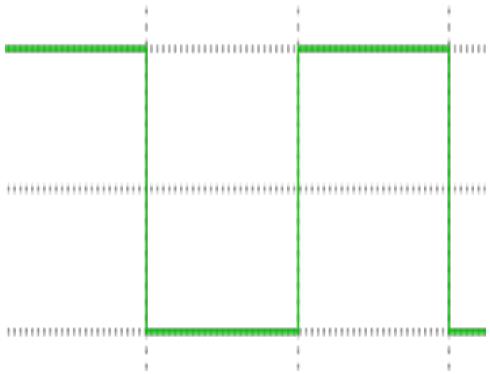
+

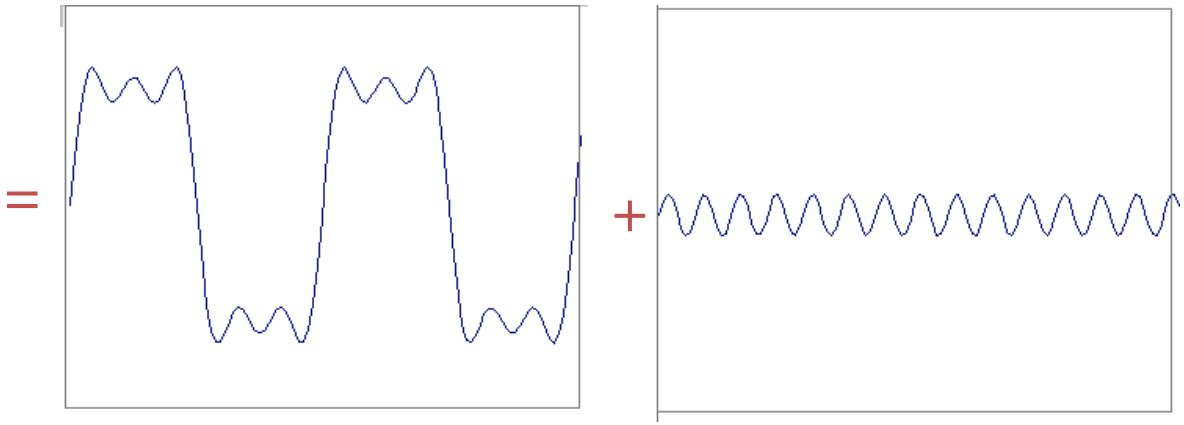


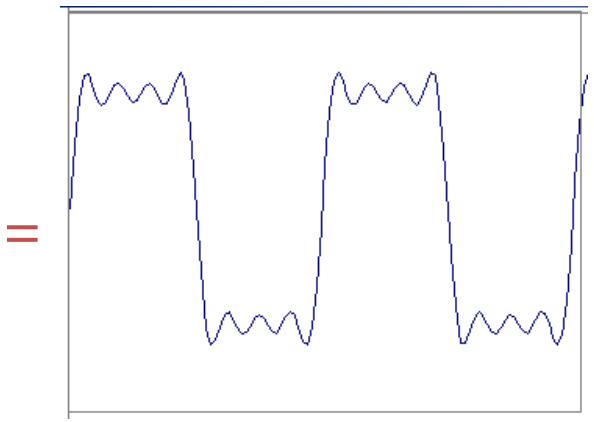
=



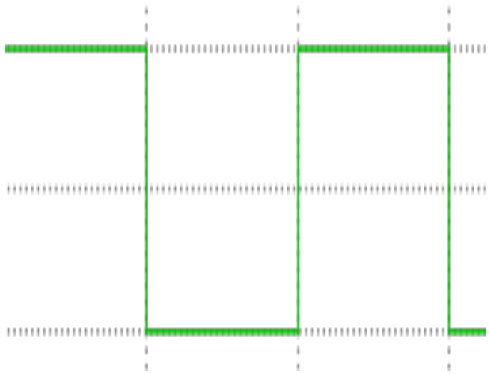
Square wave spectra

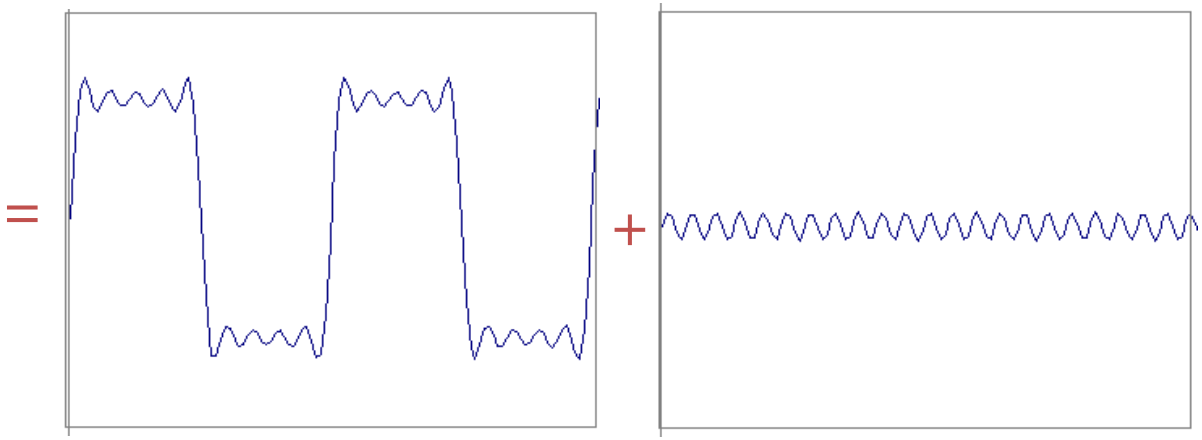


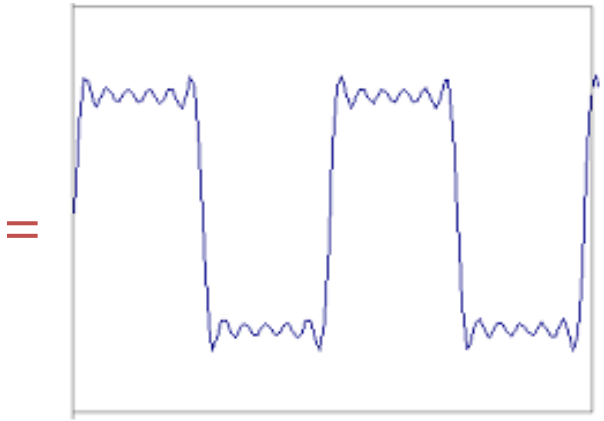




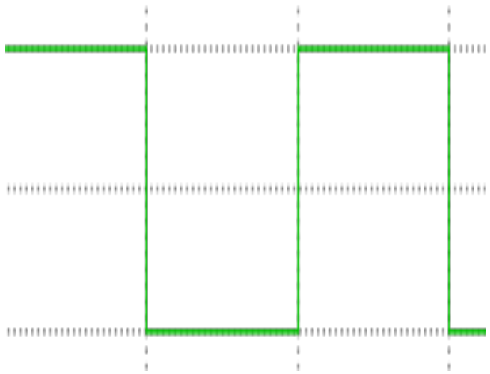
Square wave spectra

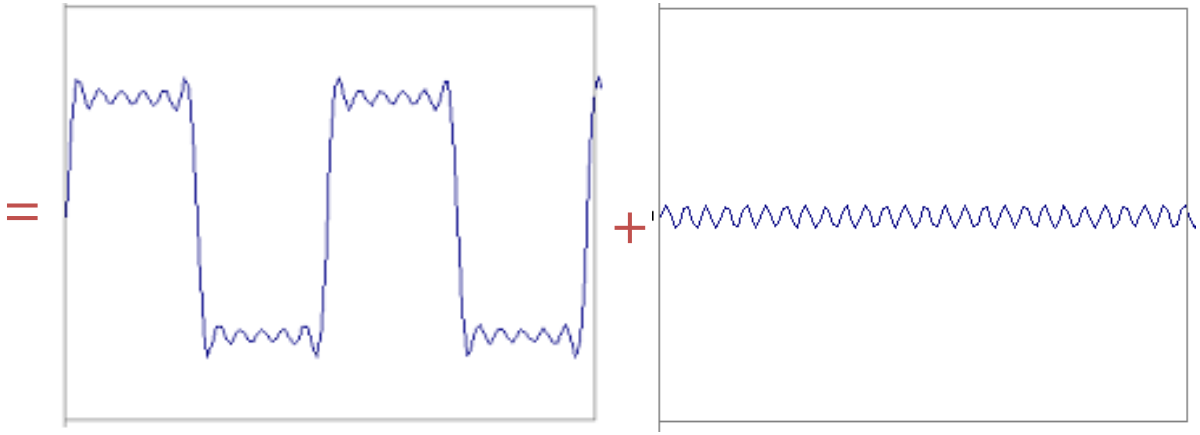


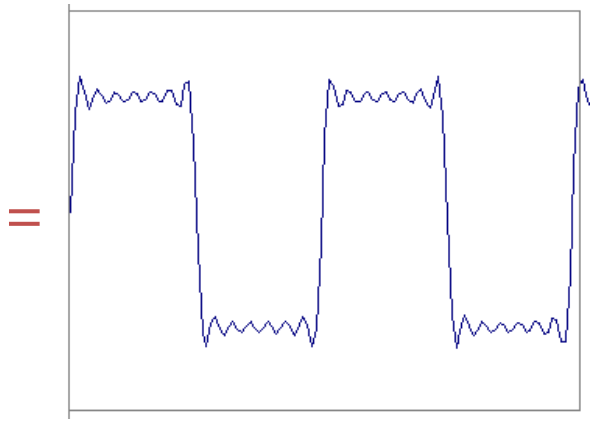




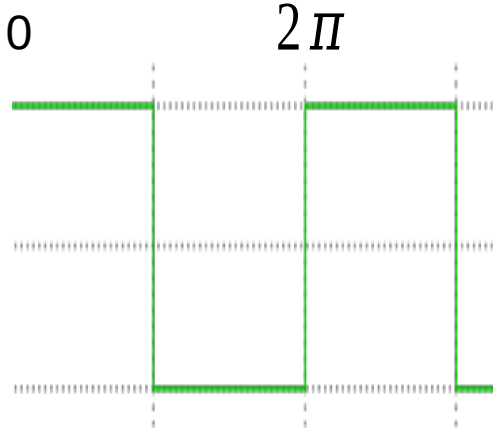
Square wave spectra





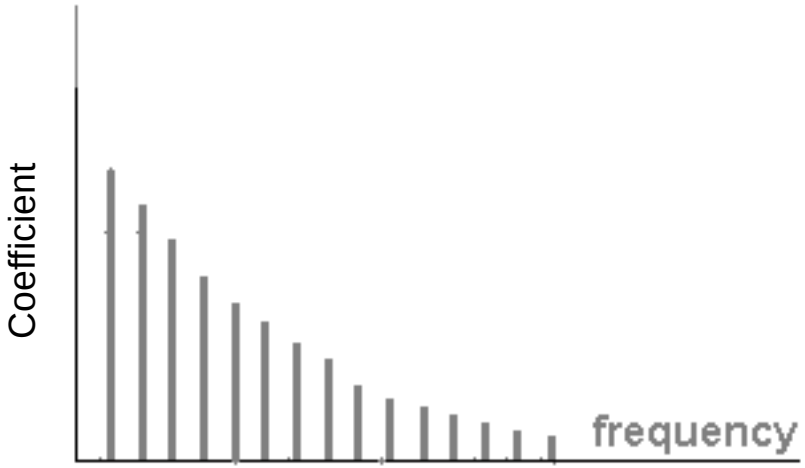


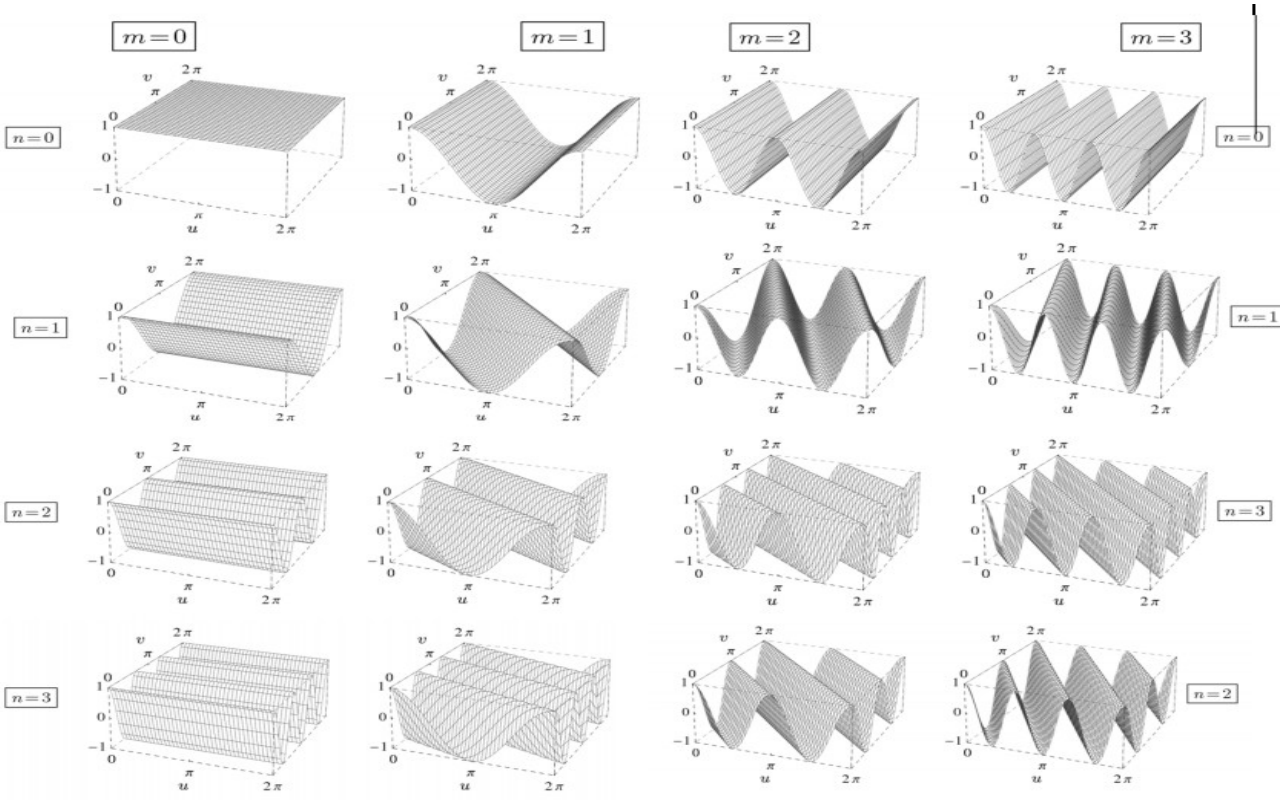
Square wave spectra



$$= \sum_{n=1}^{\infty} \frac{1}{n} \sin(nt)$$

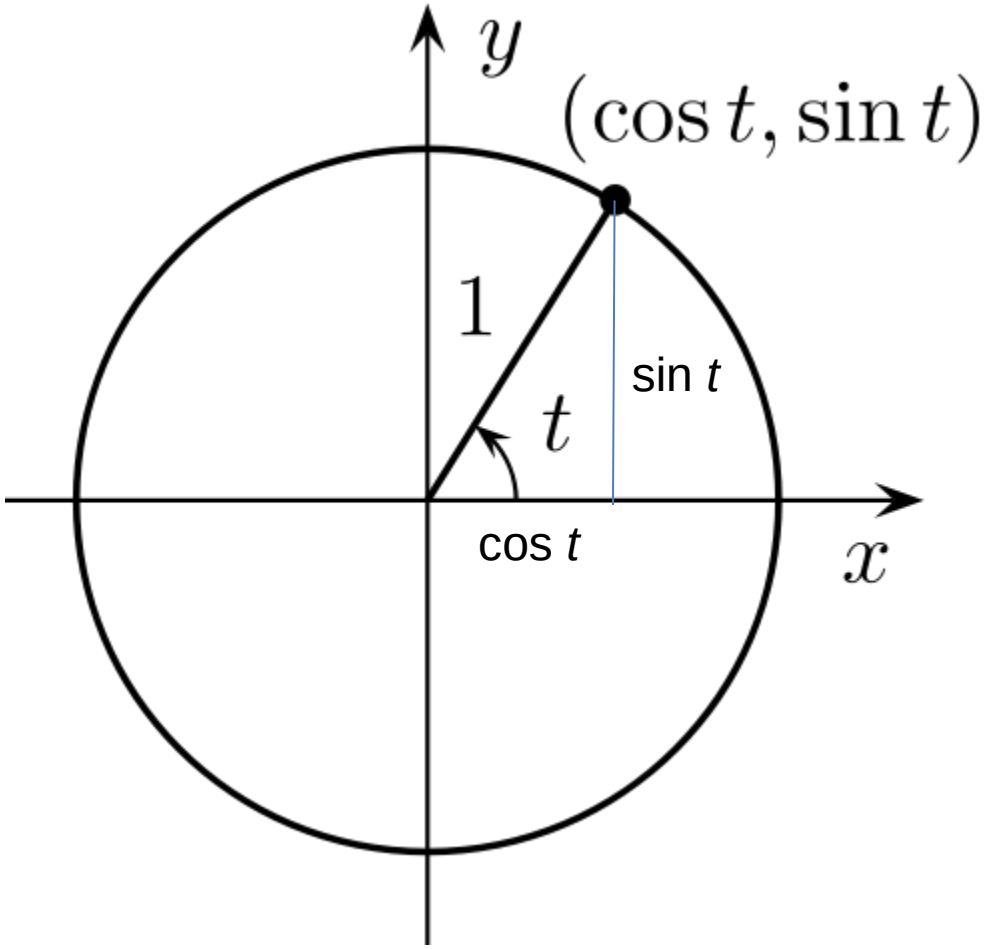
For periodic signals defined over [0,]





Wikipedia – Fourier transform

Sine/cosine and circle



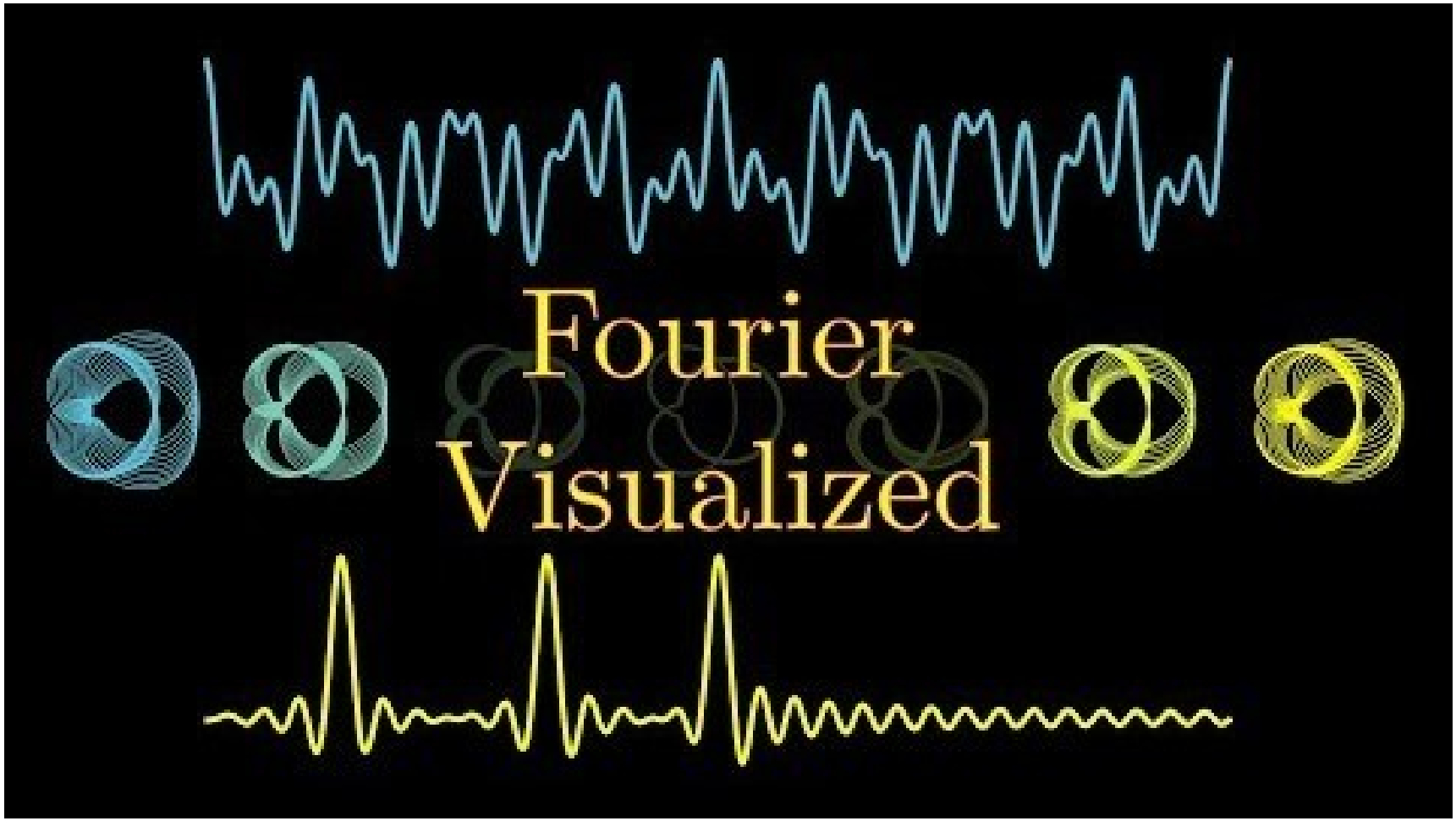
Wikipedia – Unit Circle

Square wave (approx.)

One series in each of x and y

Generative Art, じゃがりきん, Video, 2018

For a detailed visual derivation of Fourier transforms



https://www.youtube.com/watch?v=spUNpyF58BY

Fourier Transform many forms:

Sine-cosine form $\sum_{n=1}^{\infty} \left(a_n \cos(nt) + b_n \sin(nt) \right)$

The forms shown here are not complete and some terms of the full equations were omitted.

Euler complex exponential form
(using euler identity: $\cos(x) + i \sin(x) = e^{ix}$)

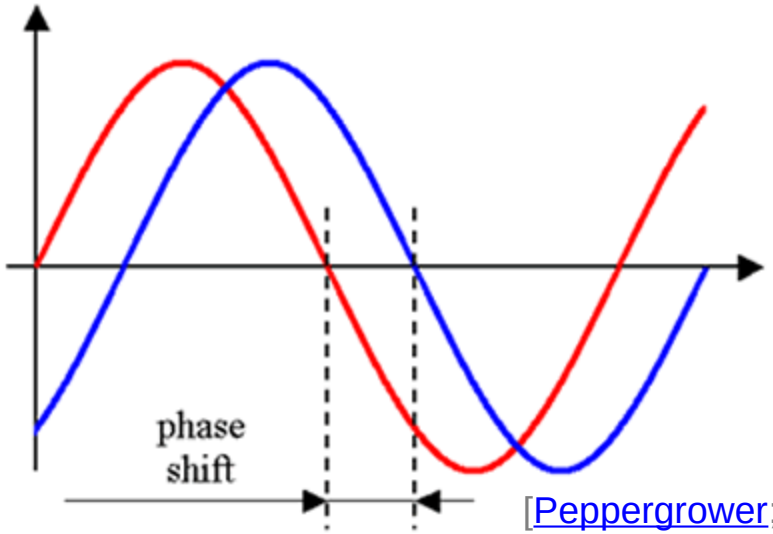
Amplitude-phase form:

Convert sine-cosine form to sine form only by converting the cosines into sines.

We do this by adding a phase term to shift the cos values into sin values

$$\sum_{n=1}^{N} \left(a_n \sin(nx + \varnothing_n) \right)$$

Phase



Morse

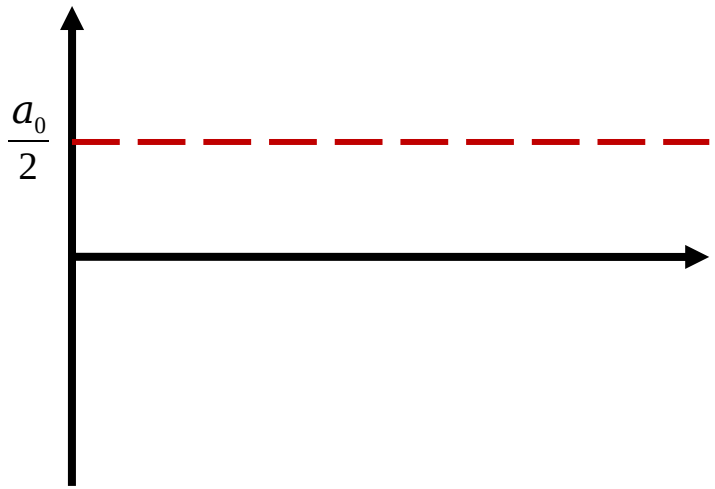
[Peppergrower; Wikipedia]

Amplitude-phase form

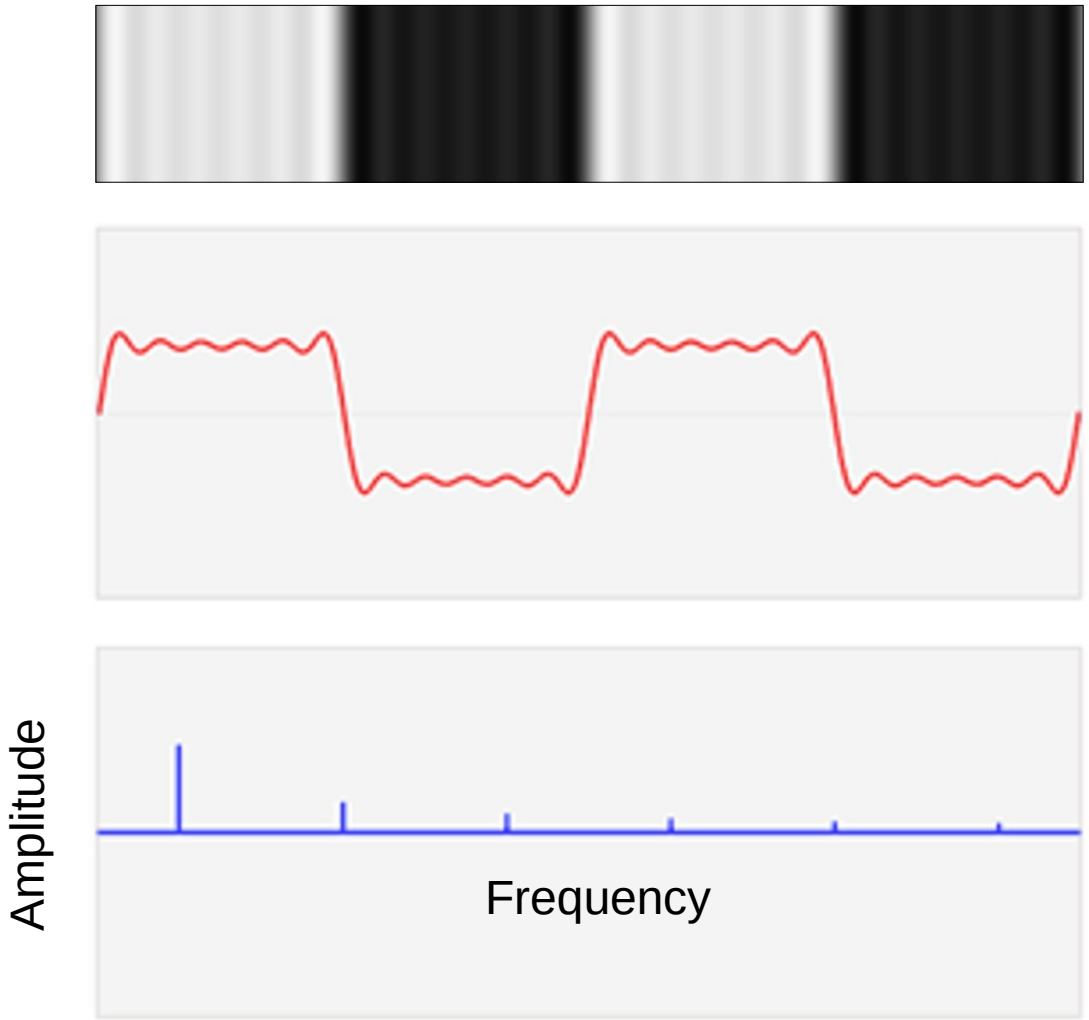
Add component of zero frequency
= mean of signal over period
= value around which signal fluctuates

$$\frac{a_0}{2} + \sum_{n=1}^{N} \left(a_n \sin\left(n x + \varnothing_n \right) \right)$$

Average of signal over period



Electronics signal processing calls this the ‘DC offset’



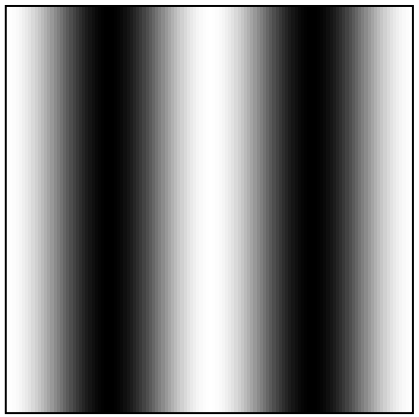
Spatial domain
imagine this as a
plane wave

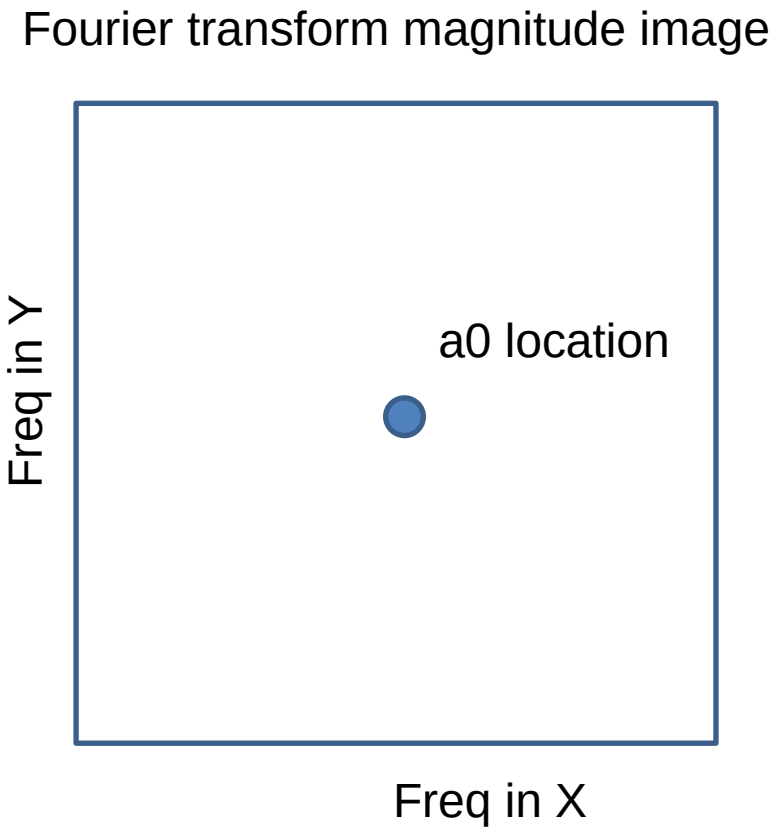
Frequency
domain

How to read Fourier transform images

We display the space such that 0-frequency coefficient is in the center (mean of pixel intensity values).

$$\frac{a_0}{2} + \sum_{n=1}^{N} \left(a_n \sin\left(n x + \varnothing_n \right) \right)$$





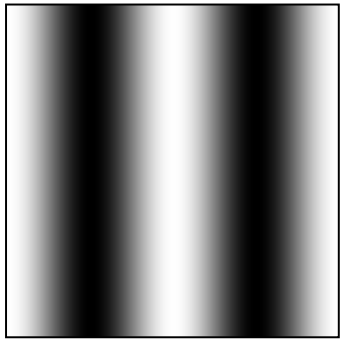
How to read Fourier transform images

Image is symmetric about center because of negative frequencies

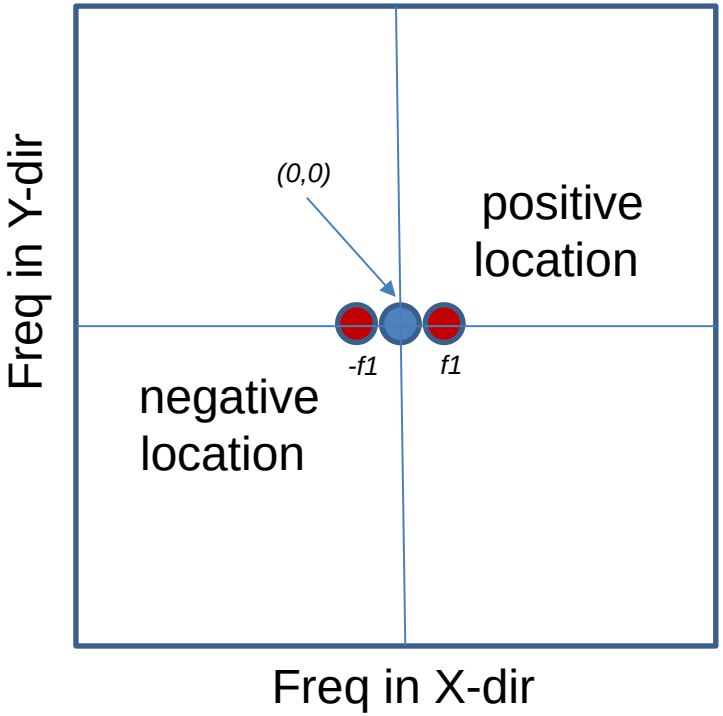
$$\frac{a_0}{2} + \sum_{n=1}^{N} \left(a_n \sin\left(n x + \varnothing_n \right) \right)$$

e.g., wheel rotating one way or the other.

For real-valued signals, positive and negative frequencies are complex conjugates



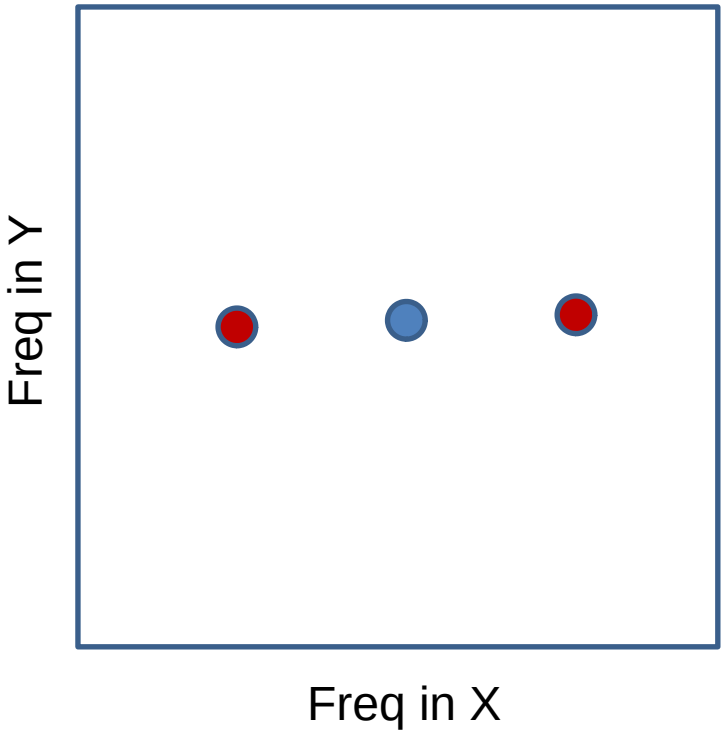
Fourier magnitude component image



How to read Fourier transform images

- Nyquist frequency is 0.5 times the sampling rate of the signal (this is not Nyquist-Shannon rate).
- In images we are sampling at every pixel then by definition the Nyquist frequency for images is 0.5 cycle/pixel.
- Nyquist frequency corresponds to the smallest element (highest frequency) one can record from a given camera system.
- If the measured signal has a higher frequency than the Nyquist frequency it causes the ‘moiré’ effect we saw.

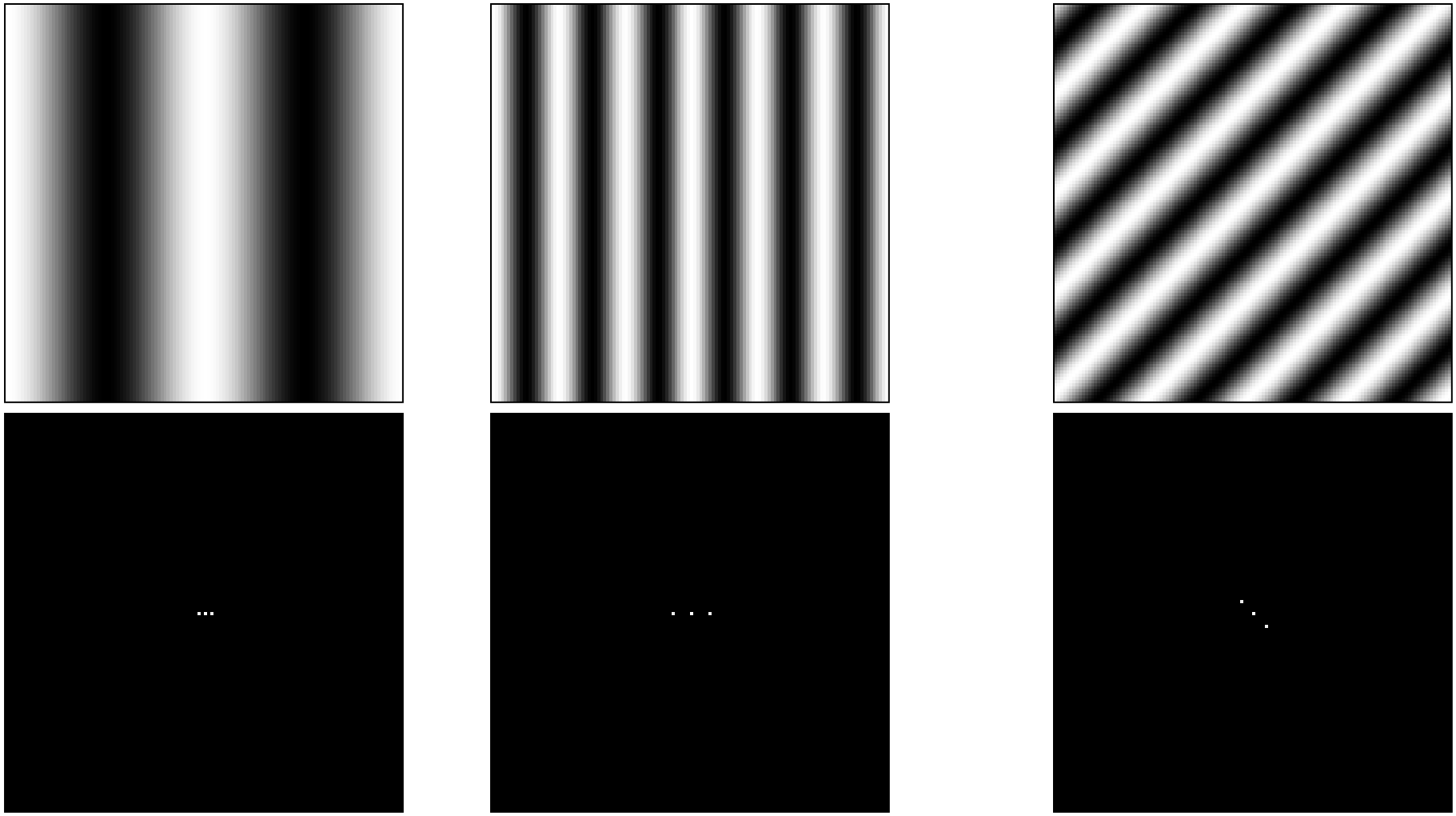
Fourier transform magnitude image



The way we can interpret this: if the point in the fourier image is half way between the center and the edge, it represents a frequency = 0.5 of the maximum frequency (Nyquist frequency) that can be observed by the imaging system.

Fourier analysis in images

Spatial domain images

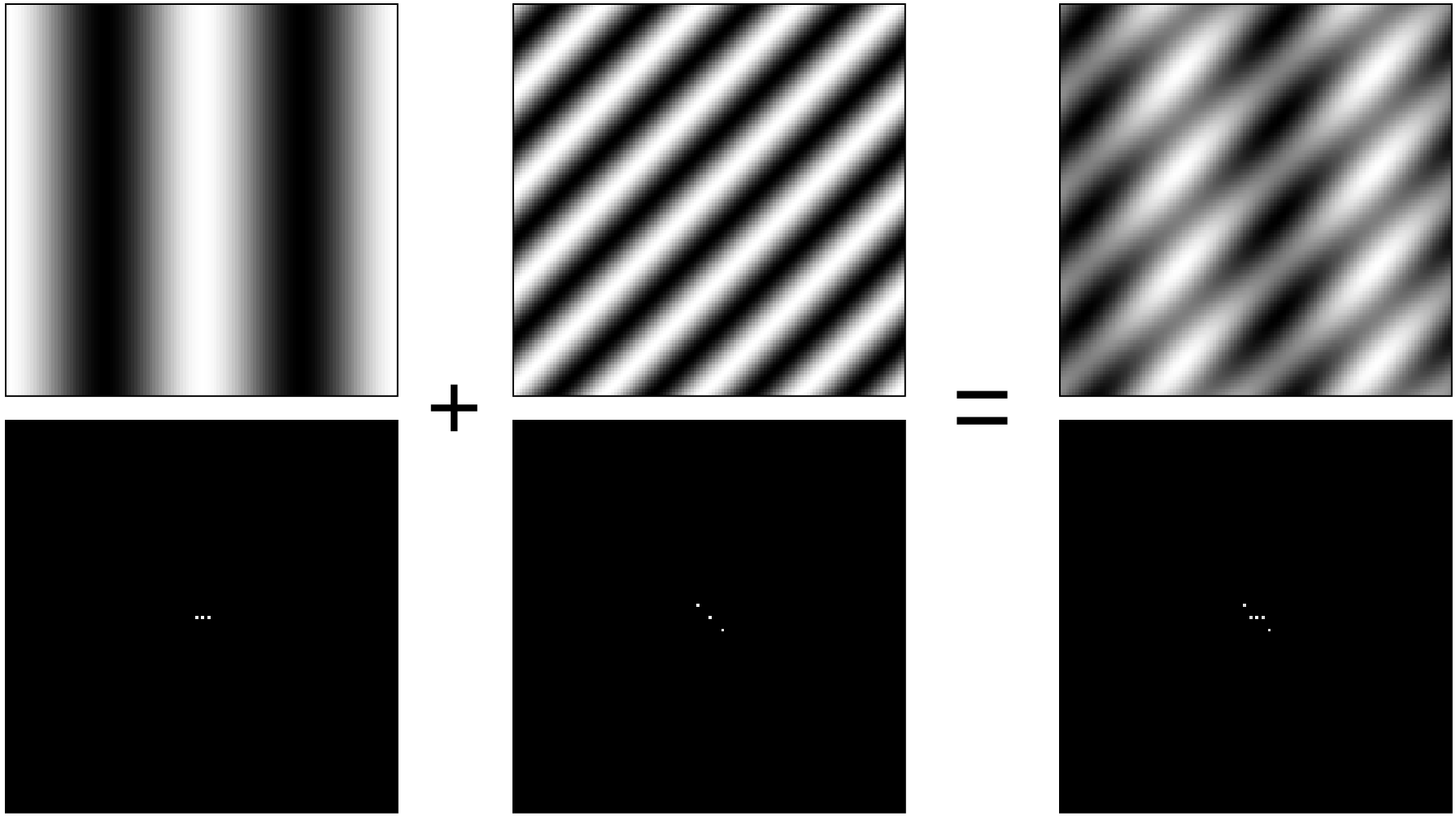


Fourier decomposition amplitude images

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

Signals can be composed

Spatial domain images



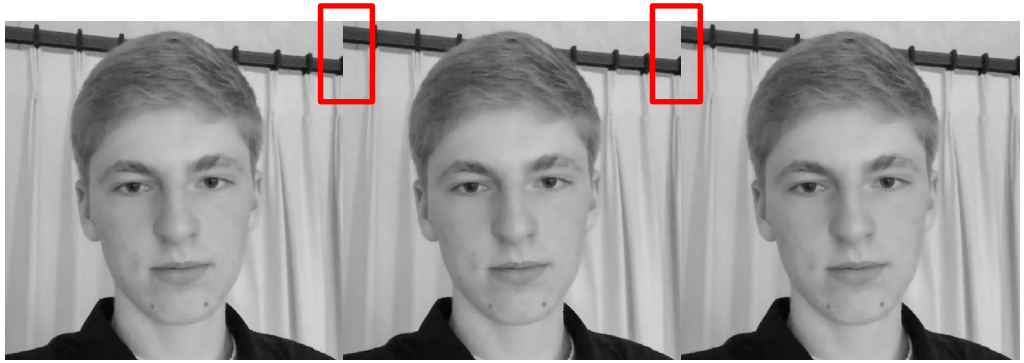
Fourier decomposition amplitude images

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

Periodicity

- Fourier decomposition assumes periodic signals with infinite extent.
  - E.G., our image is assumed to repeated forever

- ‘Fake’ signal is image wrapped around

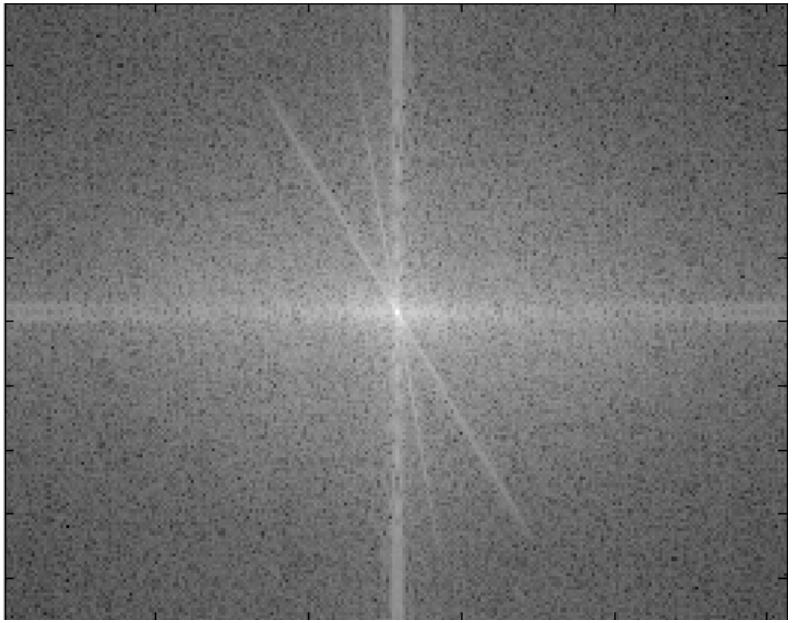


Convention is to pad with zeros to reduce effect.
We also have to specify the size of the output (typically chosen to be the same size of the original image)

Natural image

Natural image

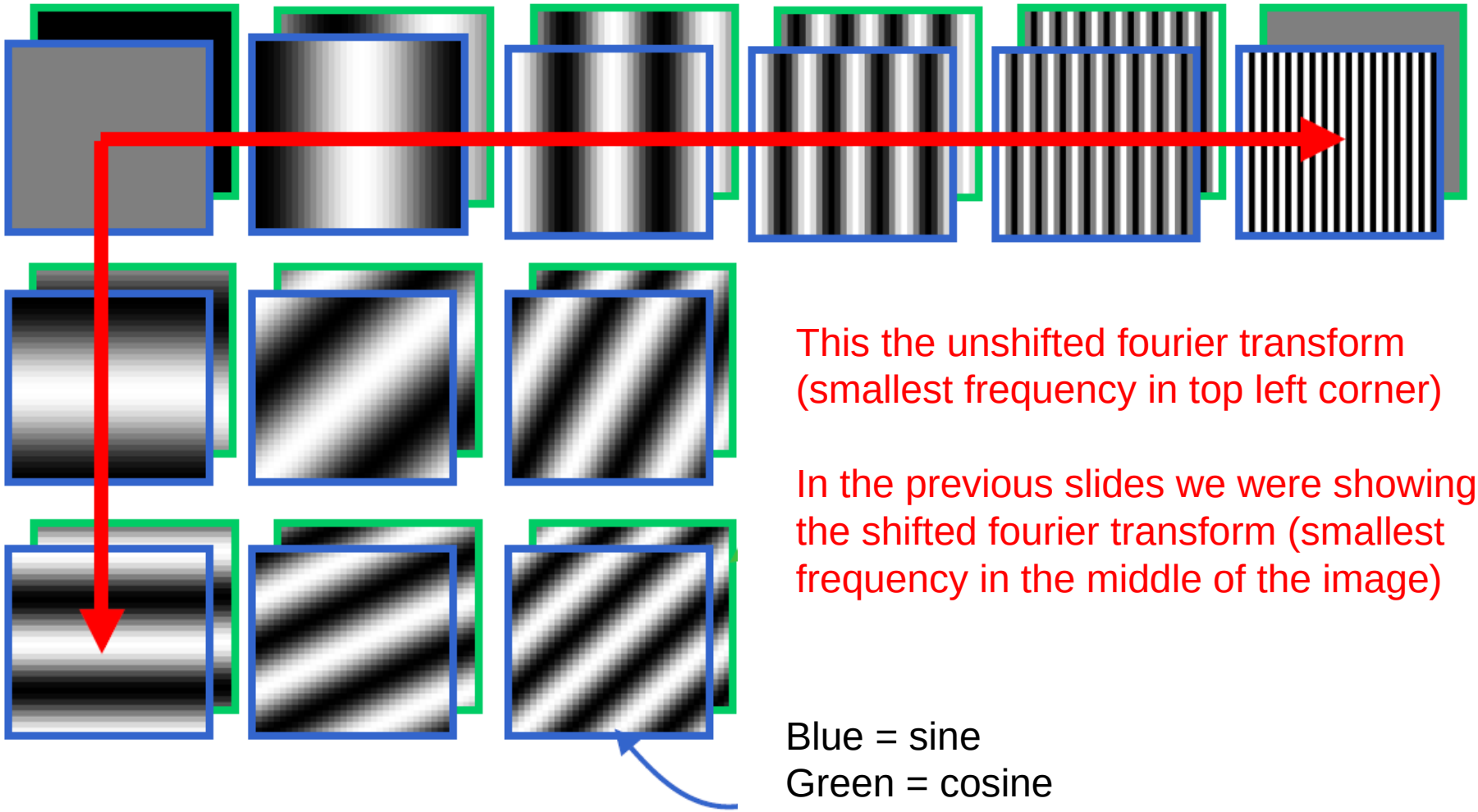
Fourier decomposition
Amplitude image



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

Teases away ‘fast vs. slow’ changes in the image.



This the unshifted fourier transform (smallest frequency in top left corner)

In the previous slides we were showing the shifted fourier transform (smallest frequency in the middle of the image)

Blue = sine
Green = cosine

This change of basis is the Fourier Transform

Basis reconstruction

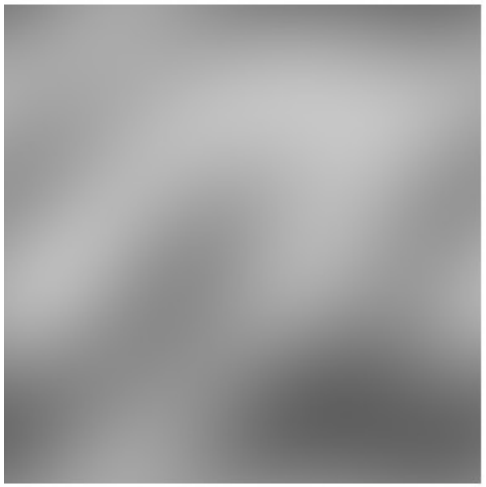
Full image

Basis reconstruction

Full image

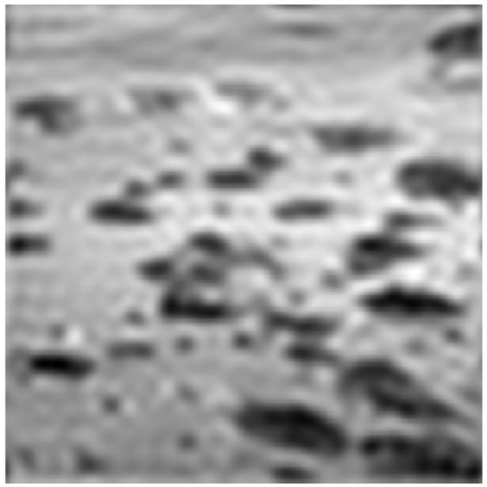
First 1 basis fn

First 4 basis fns



First 9 basis fns

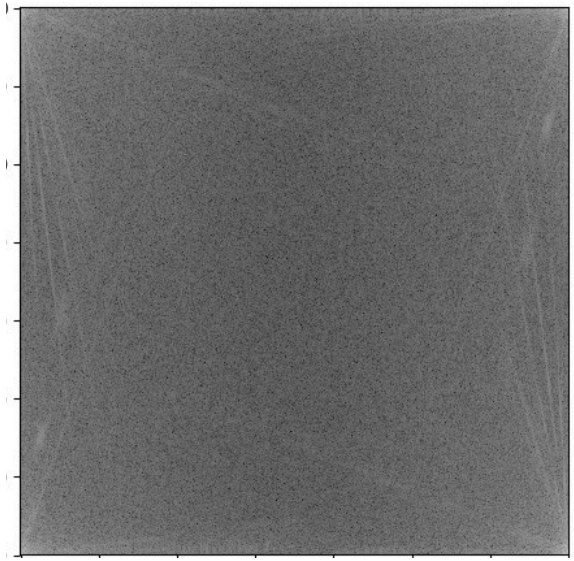
First 16 basis fns



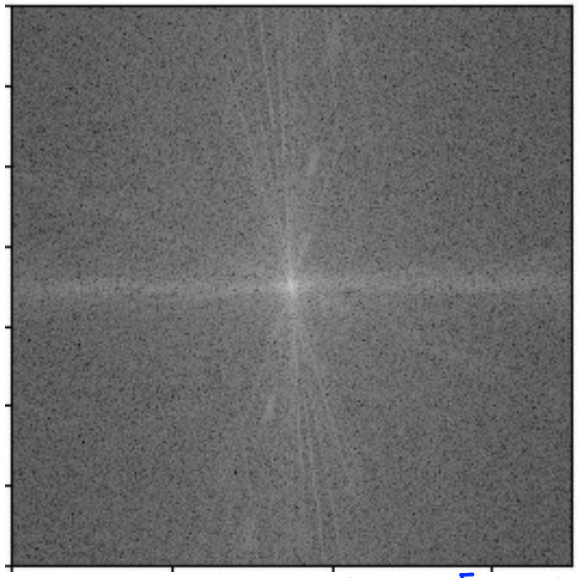
First 400 basis fns

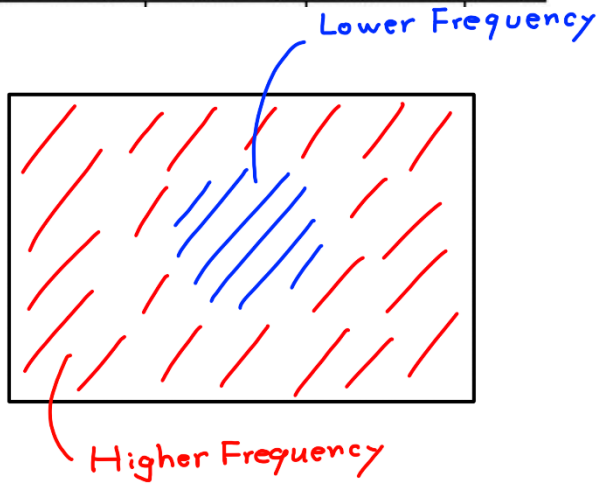
Shifted vs Unshifted Fourier transforms

Unshifted



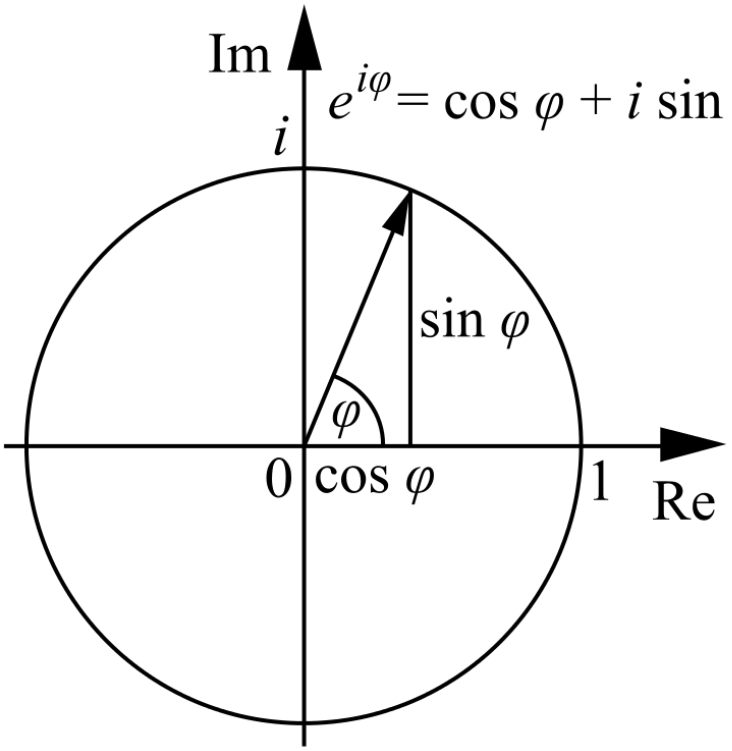
Shifted





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



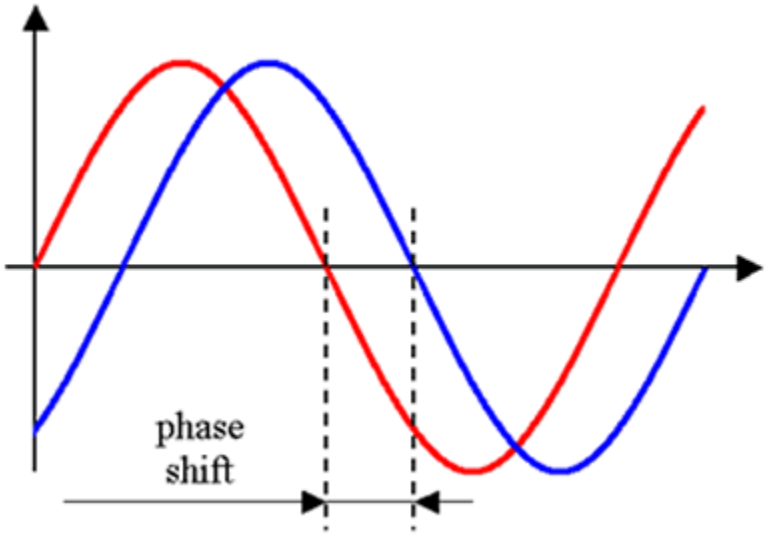
Amplitude encodes how much signal there is at a particular frequency:

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

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.

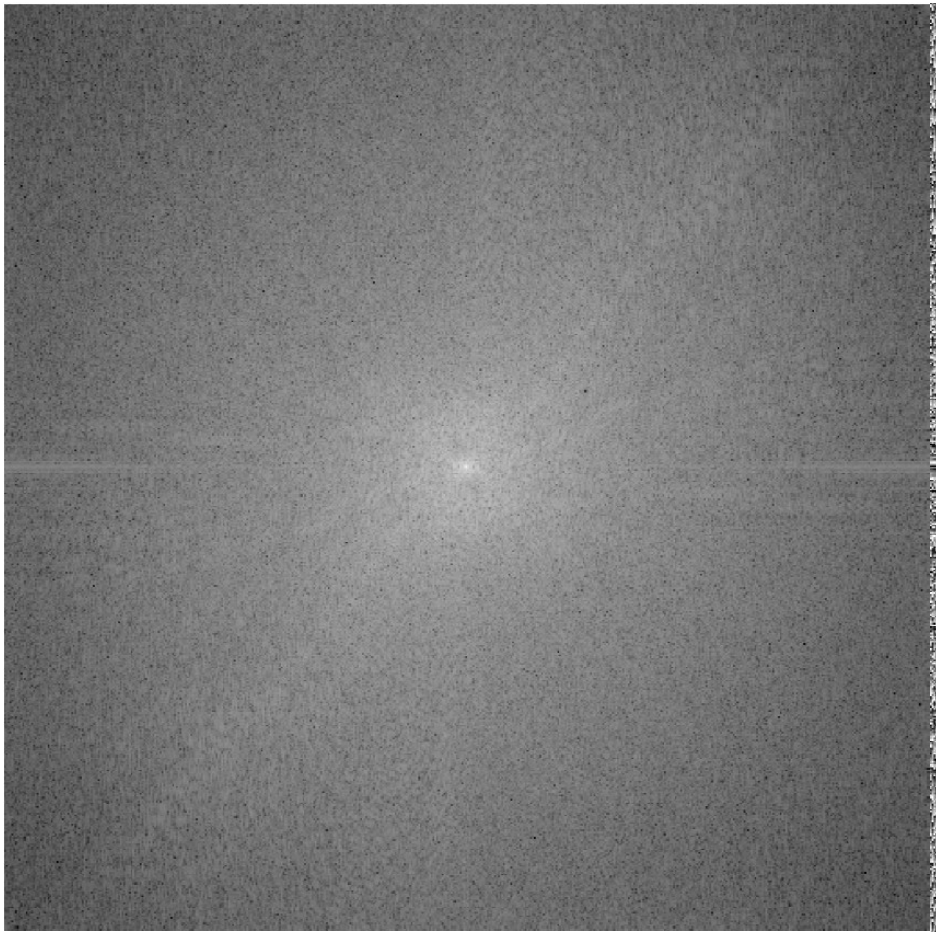
Morse

[Peppergrower; Wikipedia]

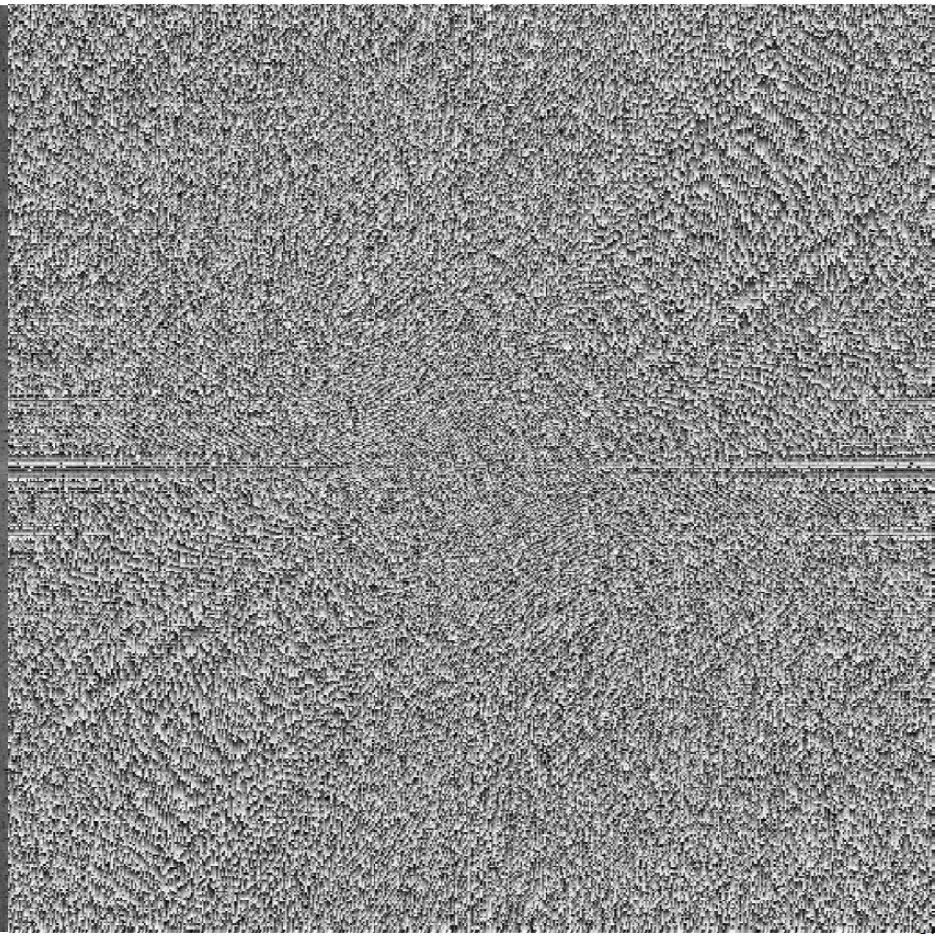
What about phase?

What about phase?

Amplitude



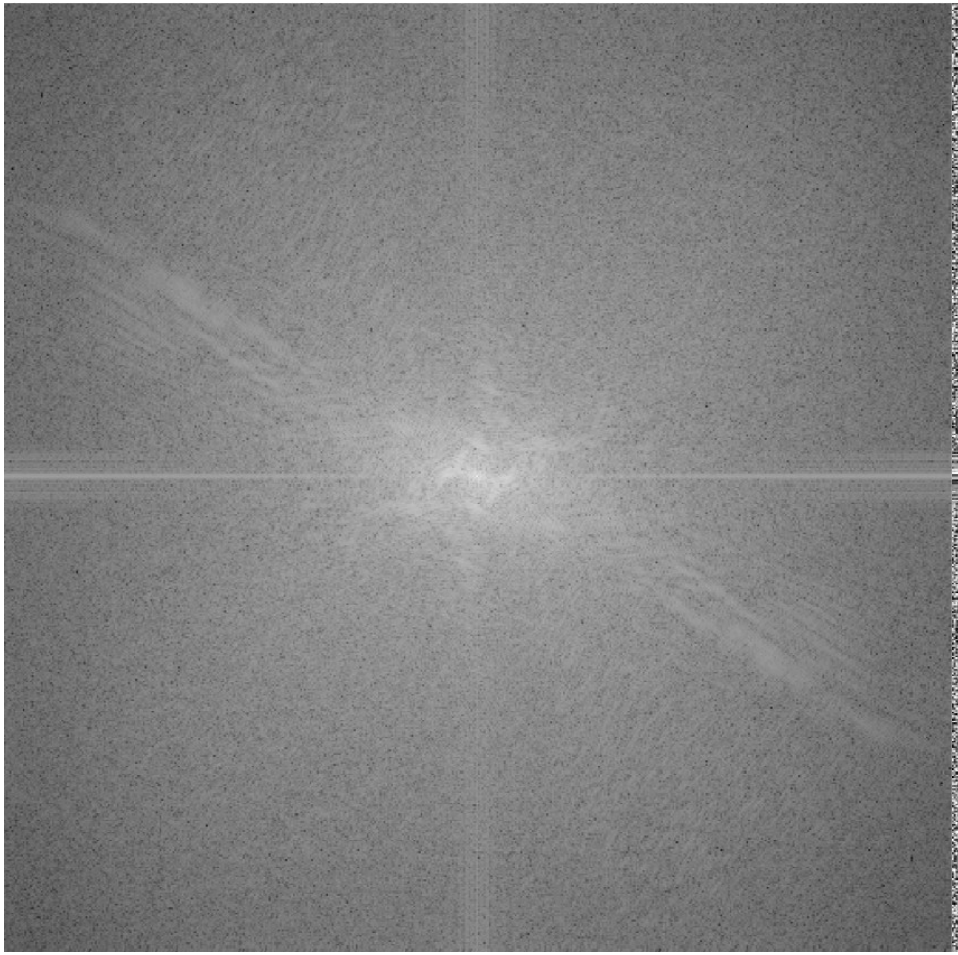
Phase



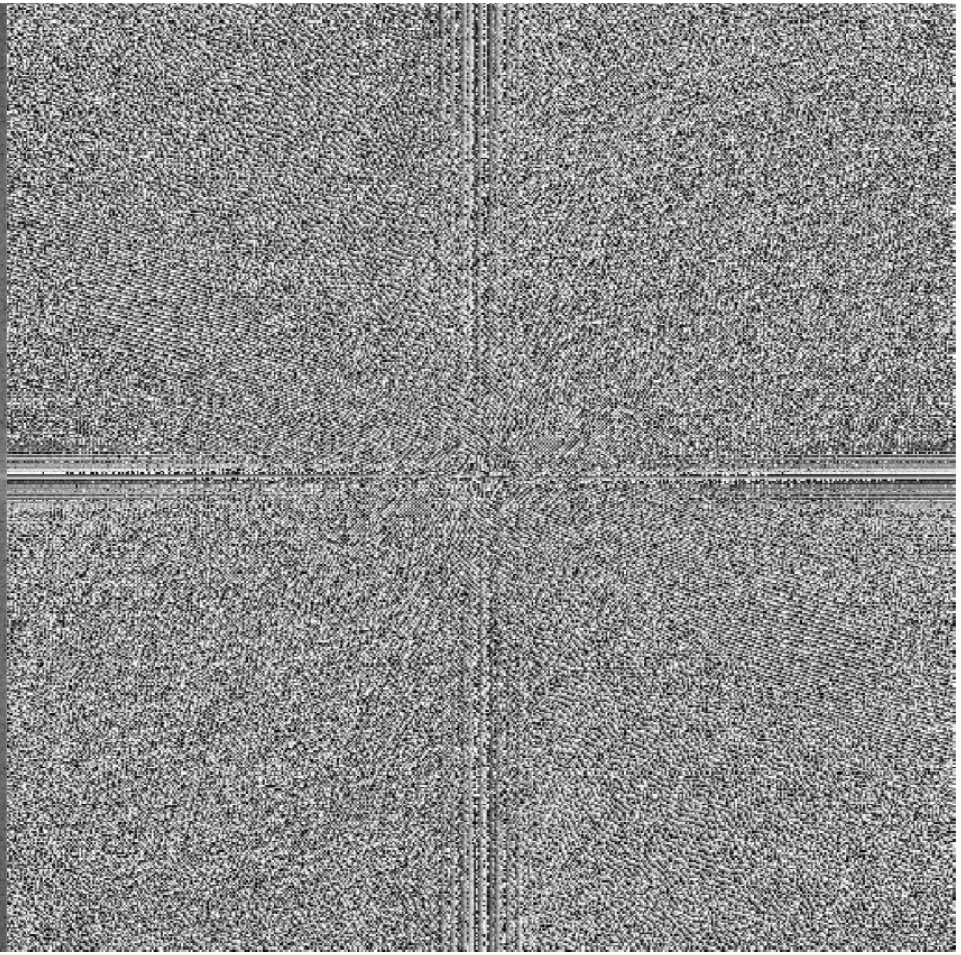
What about phase?

What about phase?

Amplitude



Phase

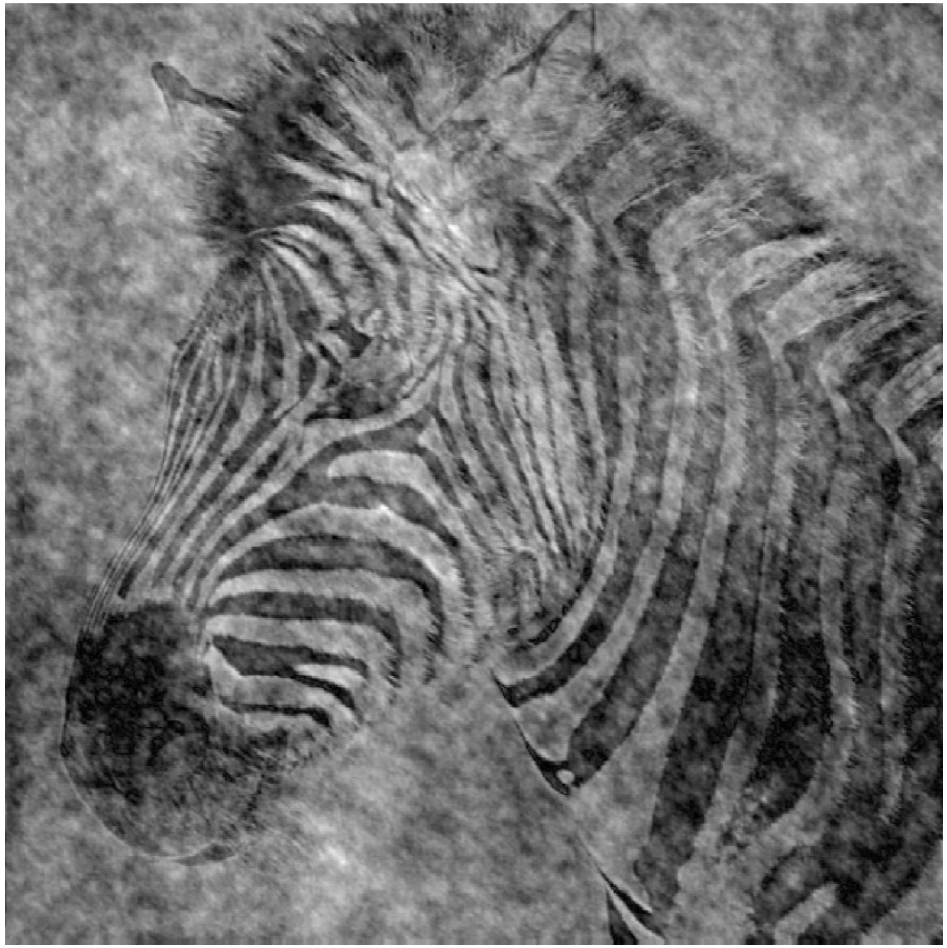


Interesting question:

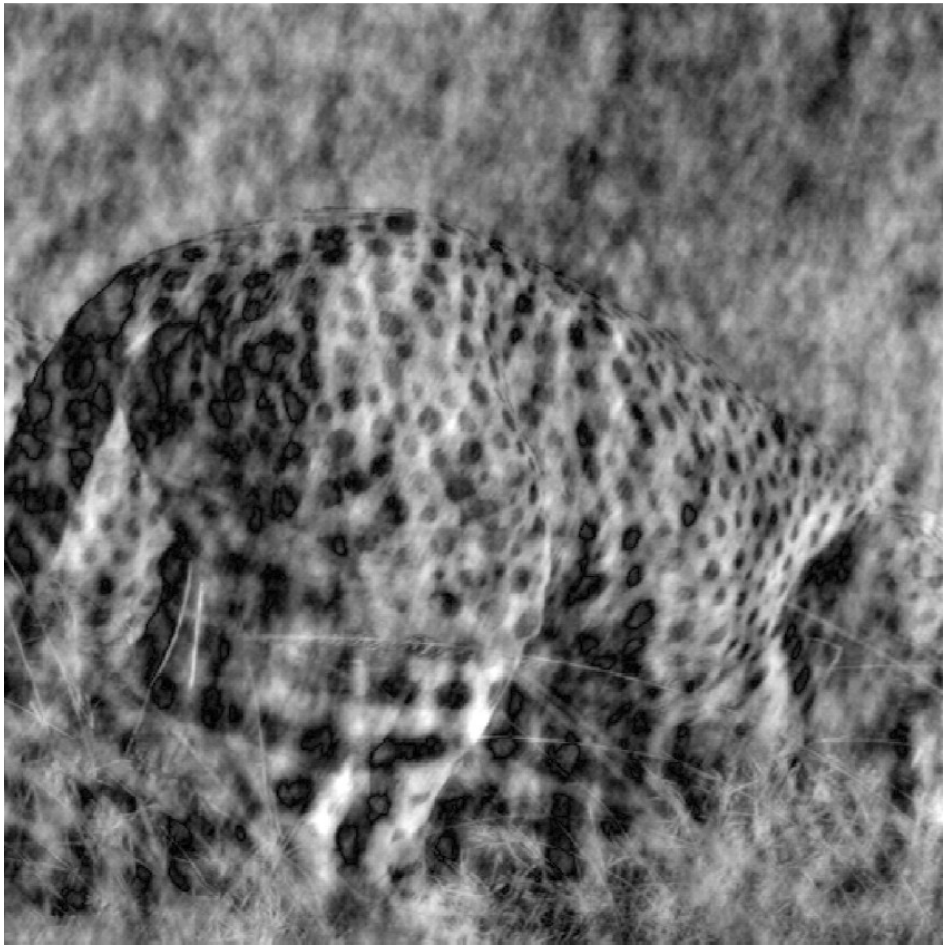
- In Fourier space, where is more of the information that we see in the visual world?
  - Amplitude
  - Phase

Cheebra

Zebra phase, cheetah amplitude



Cheetah phase, zebra amplitude



- 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

John Brayer, Uni. New Mexico

- “We generally do not display PHASE images because most people who see them shortly thereafter succumb to hallucinogenics or end up in a Tibetan monastery.”

- https://www.cs.unm.edu/~brayer/vision/fourier.html

Brian Pauw demo

- Live Fourier decomposition images
  - Using FFT2 function

Real-time fourier transforms of different shapes

https://youtu.be/aiKrrGR57aI

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

See Szeliski Book (3.4)

The Convolution Theorem

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

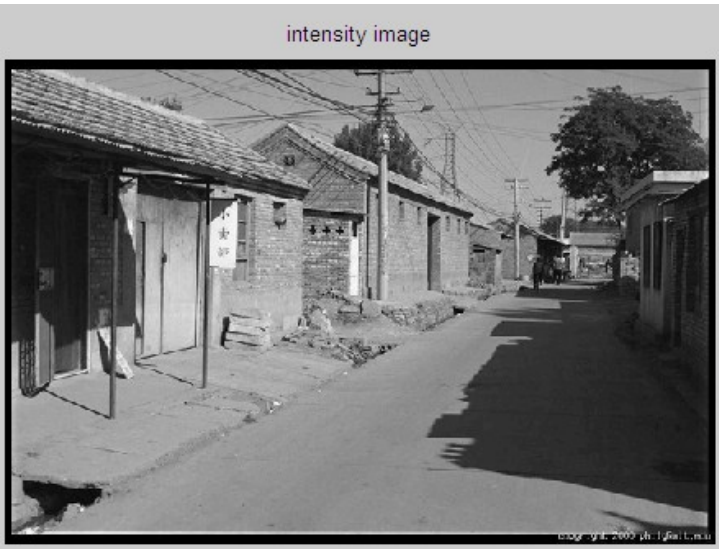
$$\mathrm{F}[g \star h] = \mathrm{F}[g]\mathrm{F}[h]$$

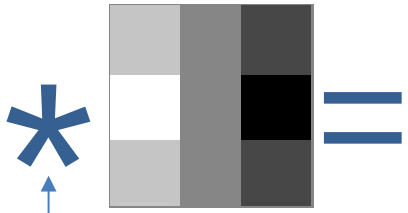
- **Convolution** in spatial domain is equivalent to **multiplication** in frequency domain!

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

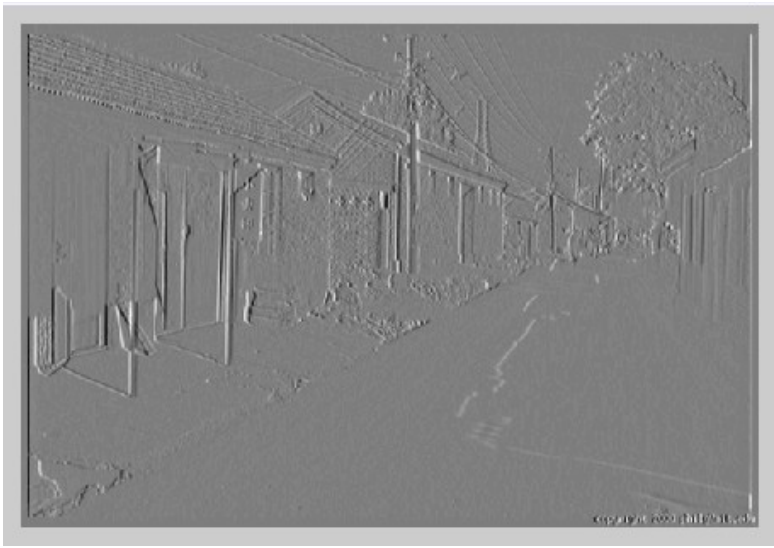
Filtering in spatial domain

1	0	-1
2	0	-2
1	0	-1

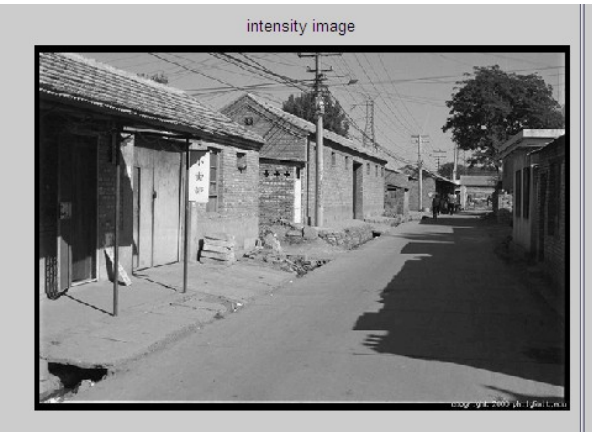




Convolution



Filtering in frequency domain

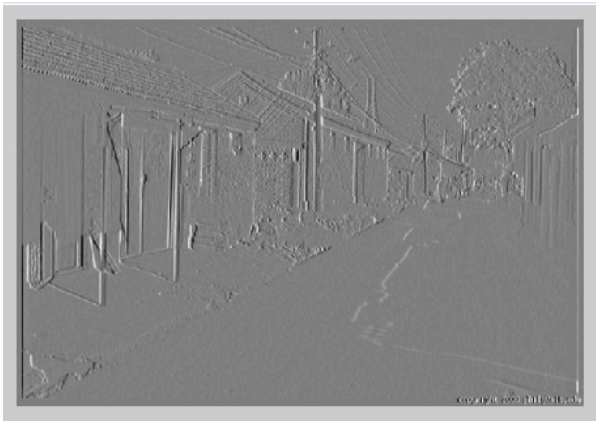


Fourier transform

Element-wise product

Fourier transform

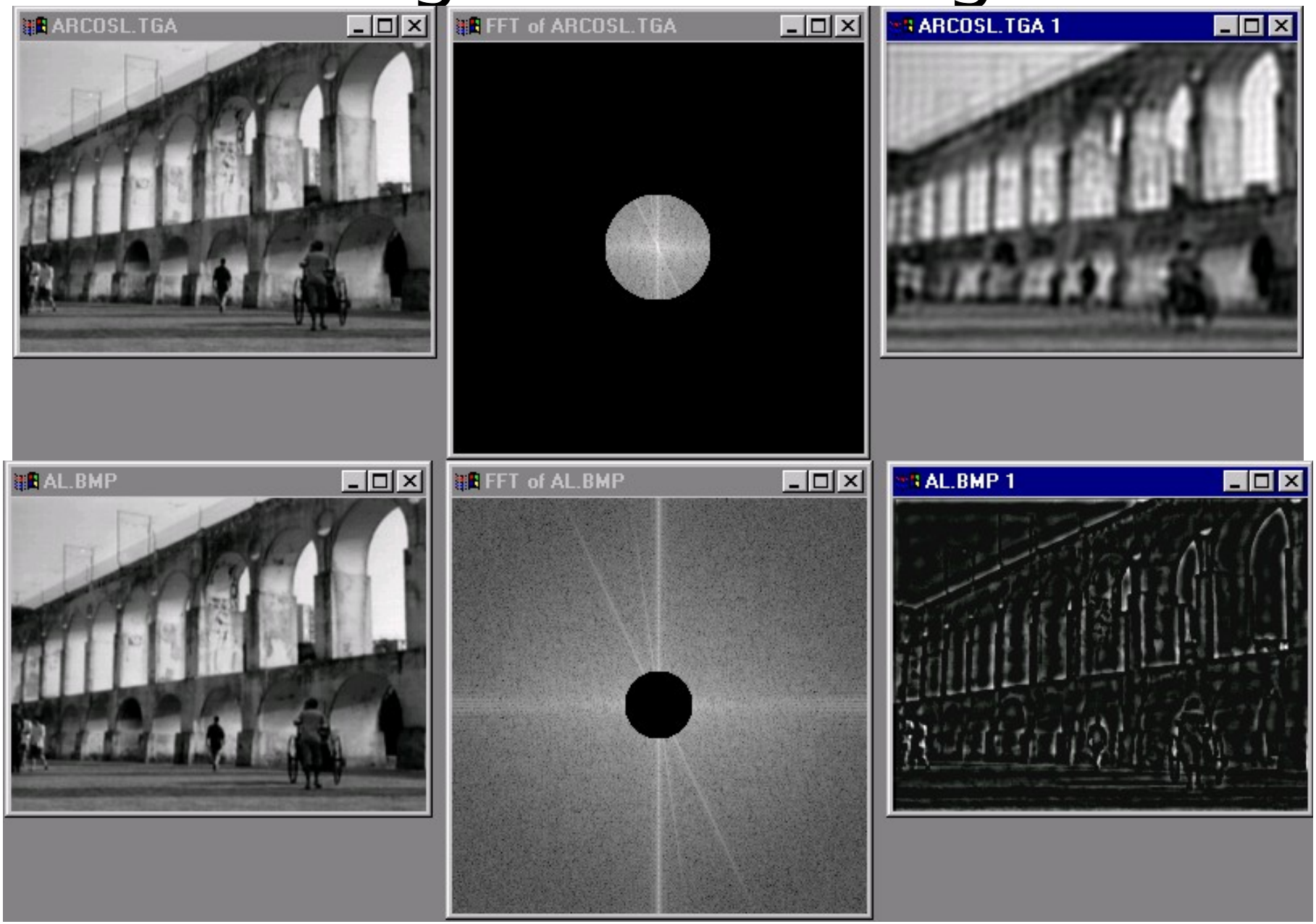
Inverse Fourier transform



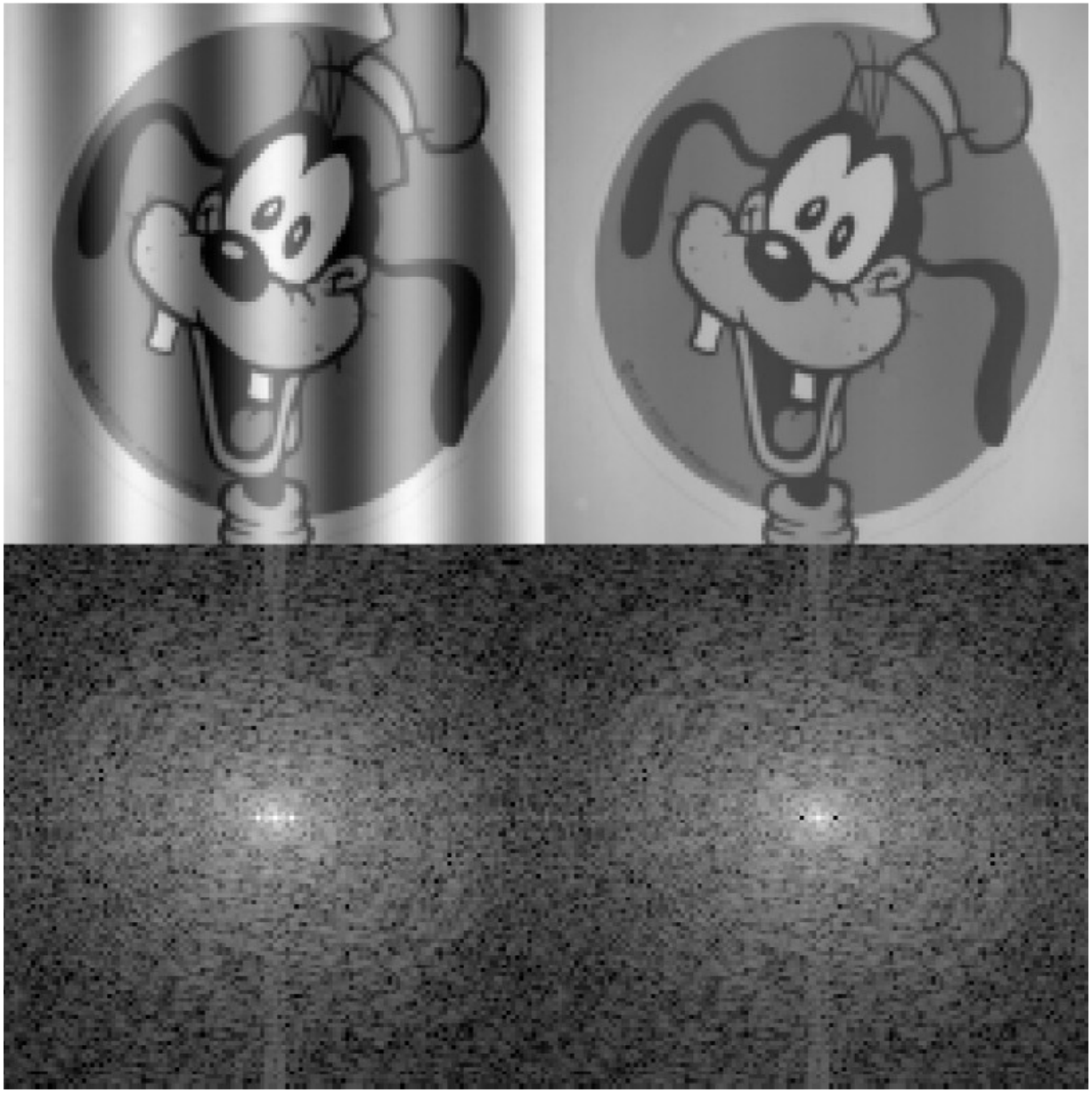
Slide: Hoiem

Now we can edit frequencies!

Low and High Pass filtering

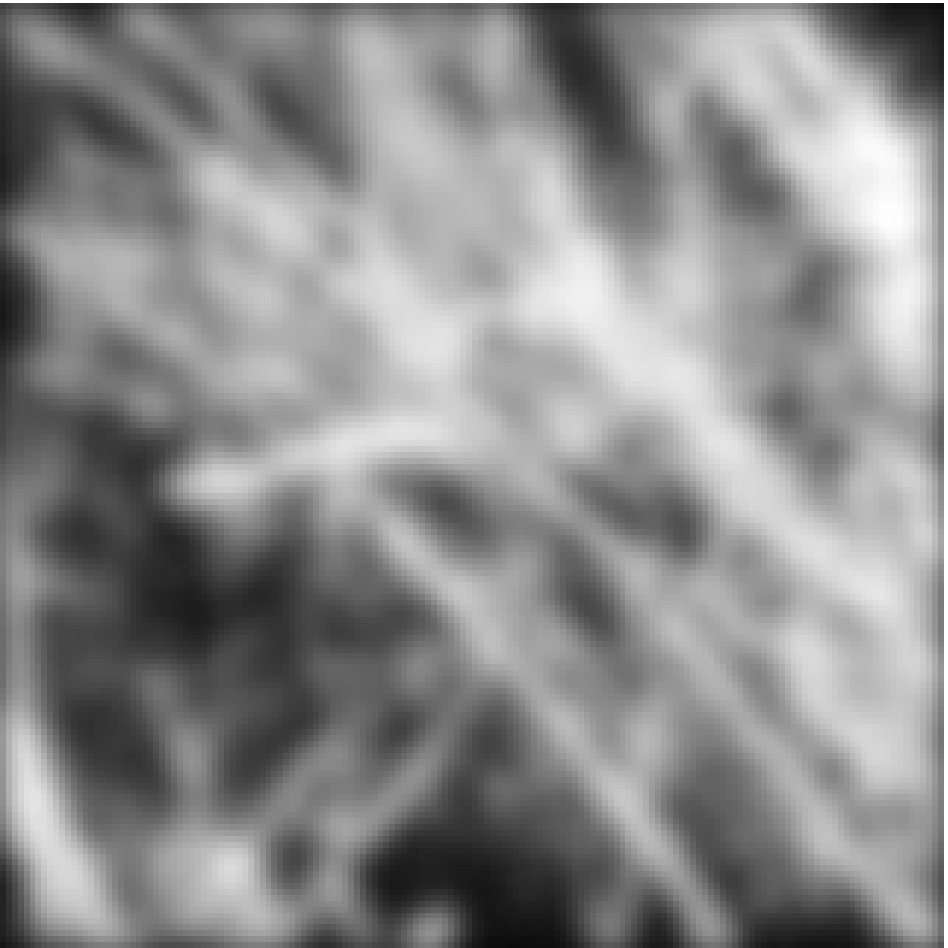


Removing frequency bands

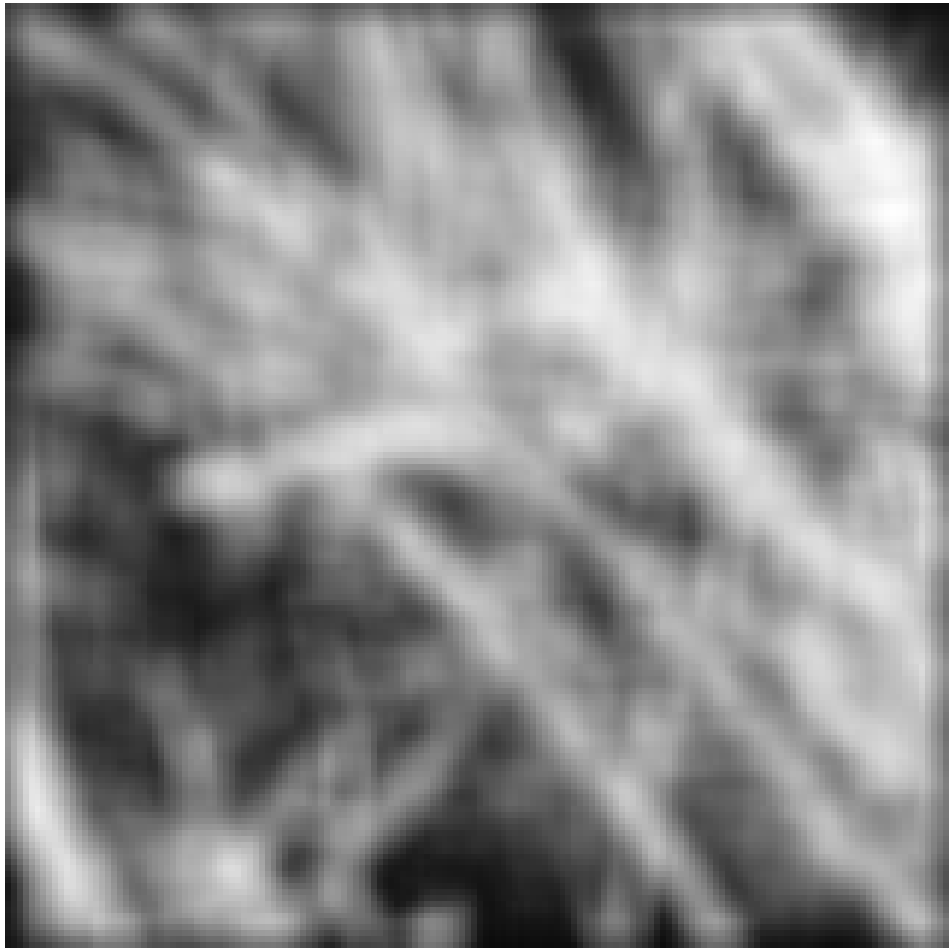


Why does the Gaussian filter give a nice smooth image, but the square filter give edgy artifacts?

Gaussian 

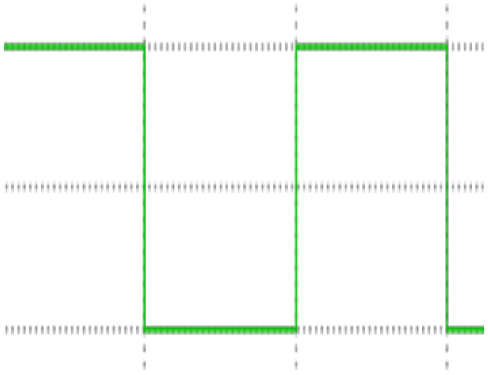


Box filter 

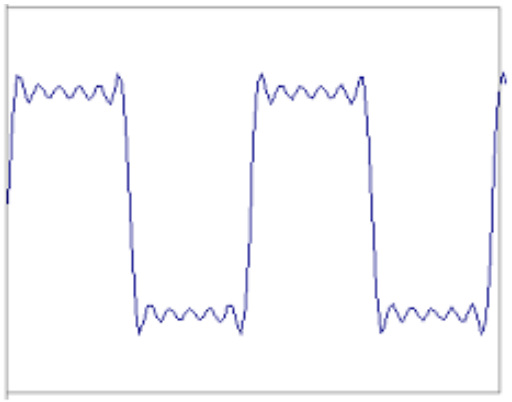


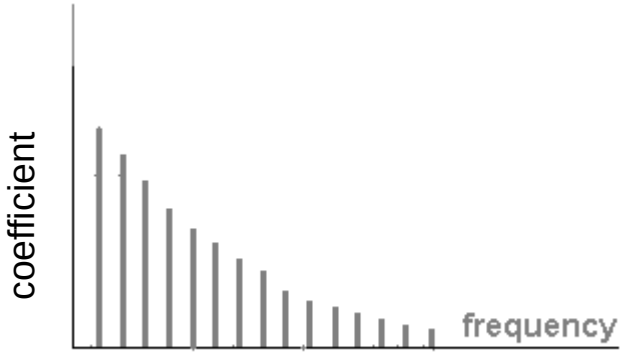
Why do we have those lines in the image?

- Sharp edges in the image need _all_ frequencies to represent them.



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

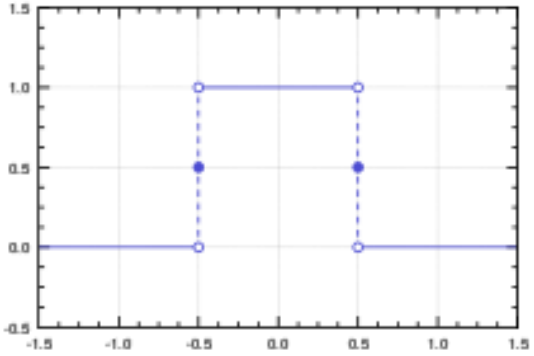




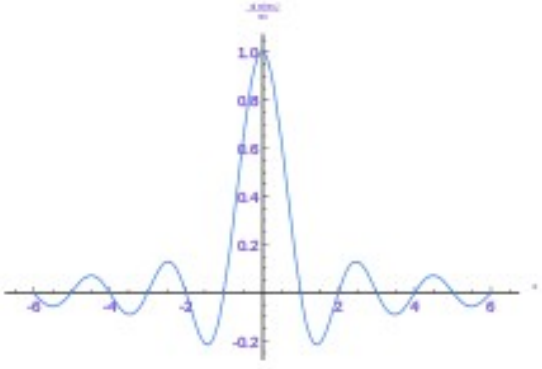
Box filter / sinc filter duality

- What is the spatial representation of the hard cutoff (box) in the frequency domain?
- http://madebyevan.com/dft/

Box filter



Sinc filter



sinc(x) = sin(x) / x

Spatial Domain ⟺ Frequency Domain

Frequency Domain ⟺ Spatial Domain

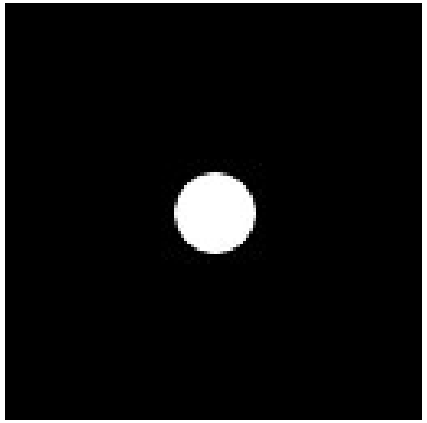
Gaussian filter duality

- Fourier transform of one Gaussian…

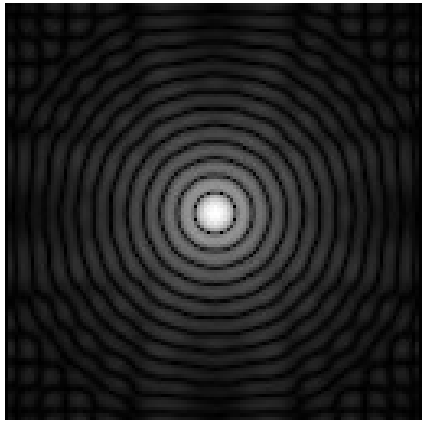
  …is *another Gaussian (with inverse variance).*

- Why is this useful?
  - Smooth degradation in frequency components
  - No sharp cut-off
  - No negative values
  - Never zero (infinite extent)

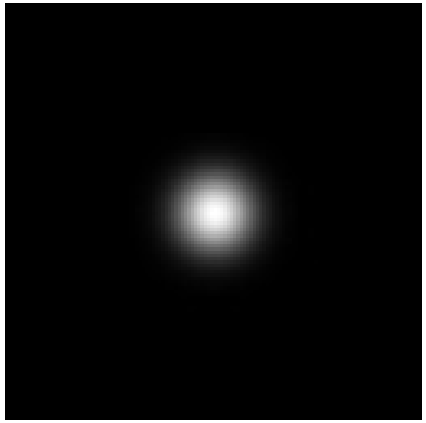
Box filter (spatial)



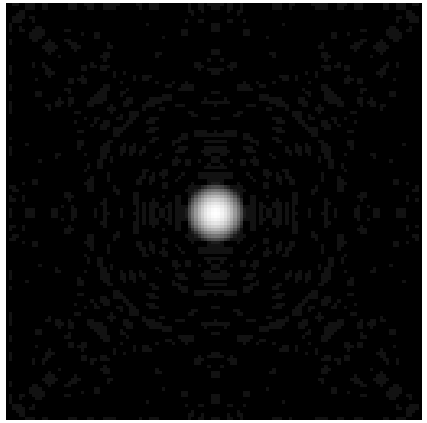
Frequency domain magnitude



Gaussian filter (spatial)



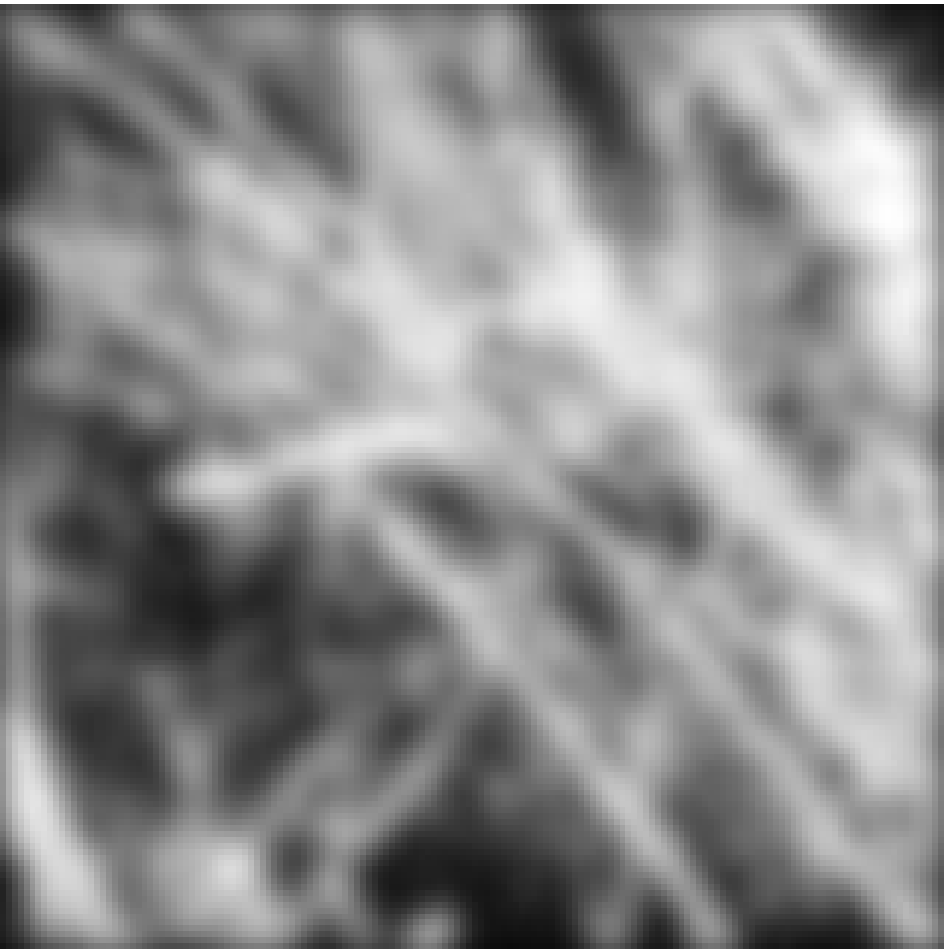
Frequency domain magnitude



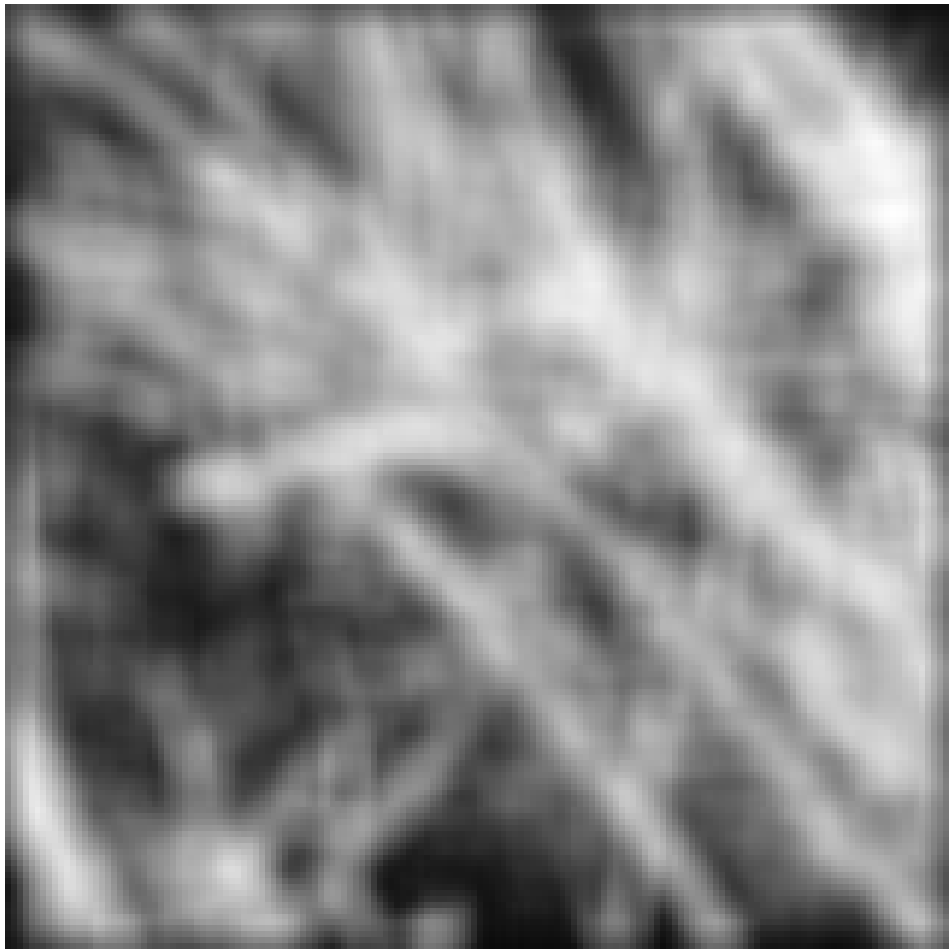
Ringing artifacts -> ‘Gibbs effect’

Where *infinite* series can never be reached

Gaussian 



Box filter 

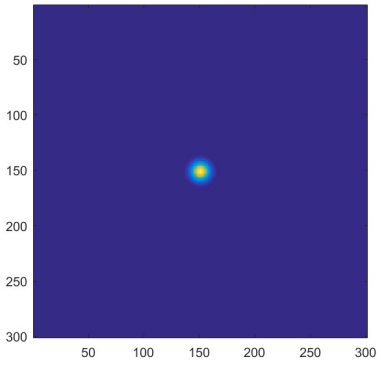


Is convolution invertible?

- 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?

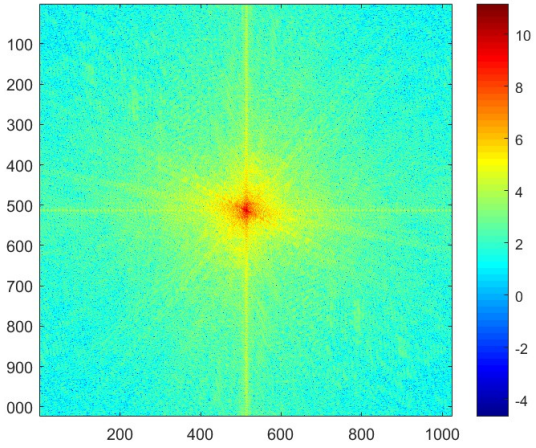
Convolution

*



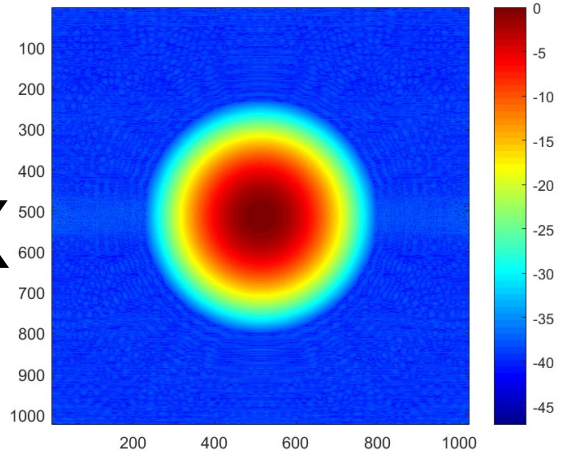
=

FFT



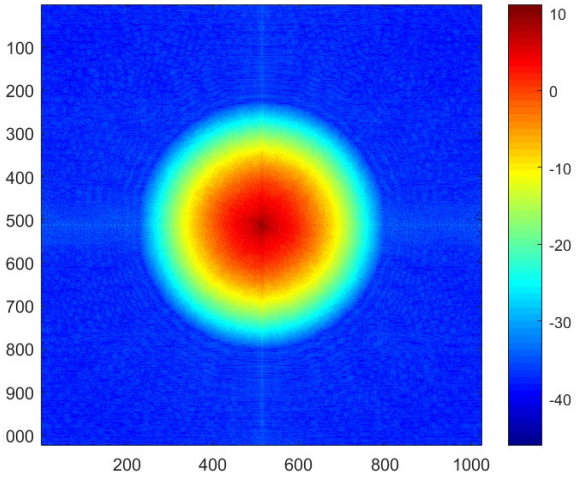
FFT

.x

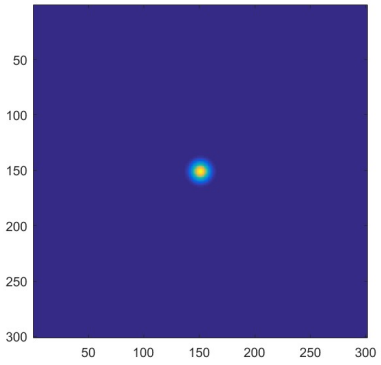


iFFT

=



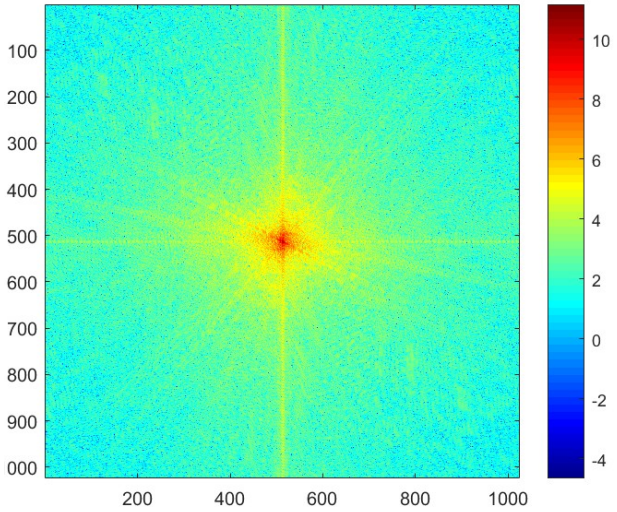
Deconvolution?



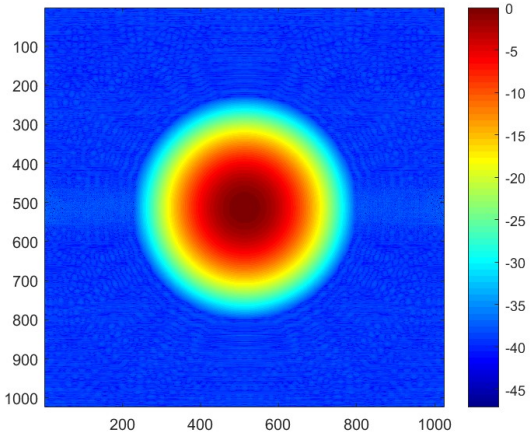
iFFT

FFT

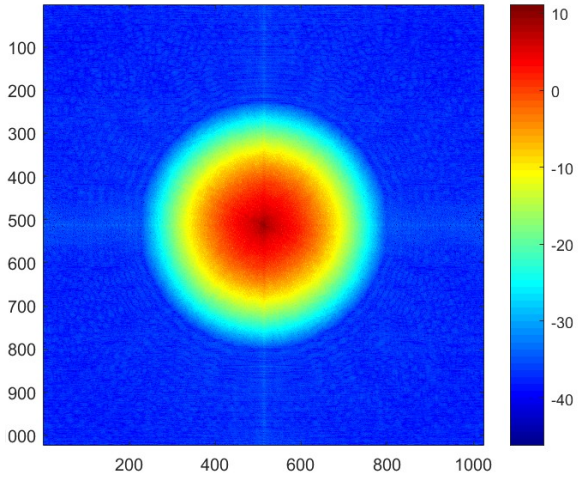
FFT



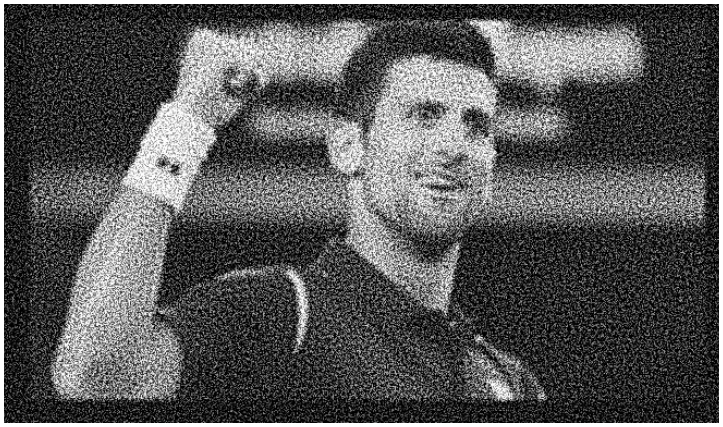
=

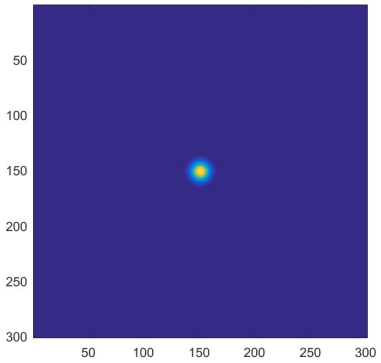


./



But under more realistic conditions



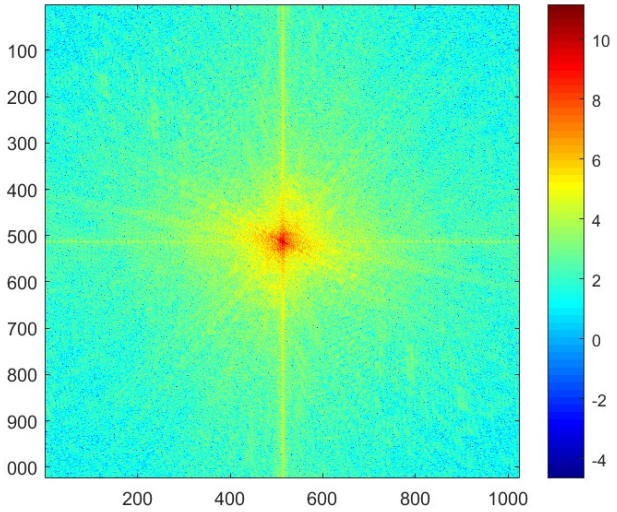


Random noise, .001 magnitude

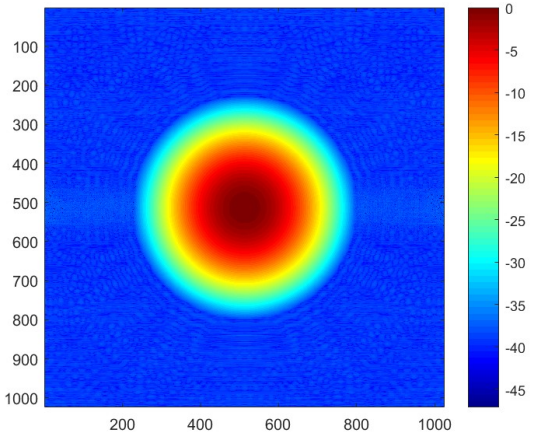
iFFT

FFT

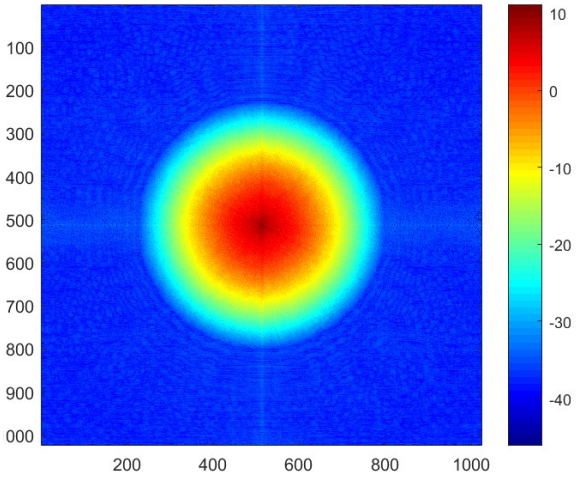
FFT



=



./



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.

A few questions

How is the Fourier decomposition computed?

Intuitively, by correlating the signal with a set of waves of increasing frequency!

There are many implementations of Fourier transforms (eg. Fast Fourier Transforms, etc.) Won’t get into the implementation details

Computing the Fourier Transform in one dimension

$$H(\omega) = \mathcal{F}\{h(x)\} = Ae^{j\phi}$$

Continuous

$$H(\omega) = \int_{-\infty}^{\infty} h(x)e^{-j\omega x}dx$$

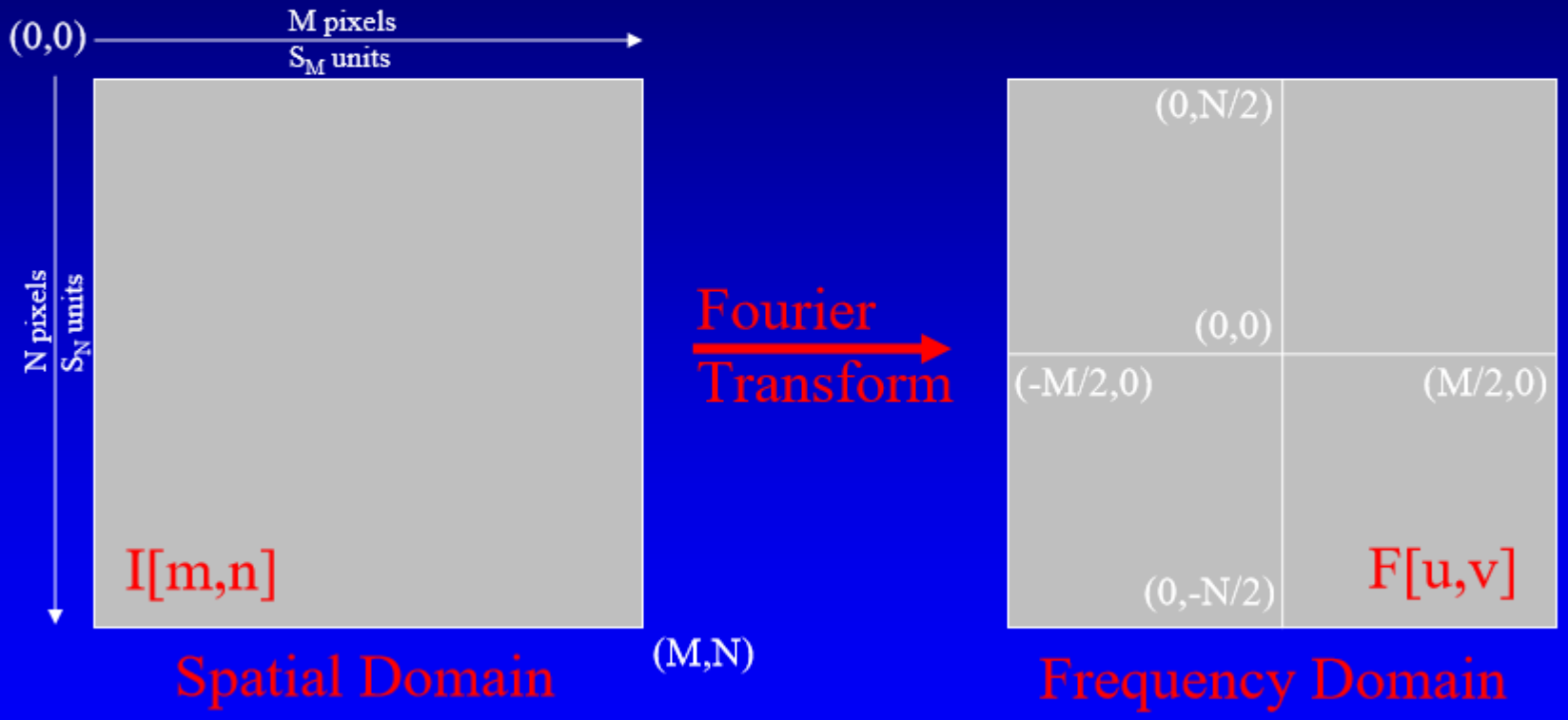
Discrete

$$H(k) = \frac{1}{N}\sum_{x=0}^{N-1} h(x)e^{-j\frac{2\pi kx}{N}}$$ *k = -N/2..N/2*

Fast Fourier Transform (FFT): NlogN

2D Discrete Fourier Transform

$$F[u,v] = \frac{1}{MN} \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} I[m,n] \cdot e^{-i2\rho\left(\frac{um}{M}+\frac{vn}{N}\right)}$$

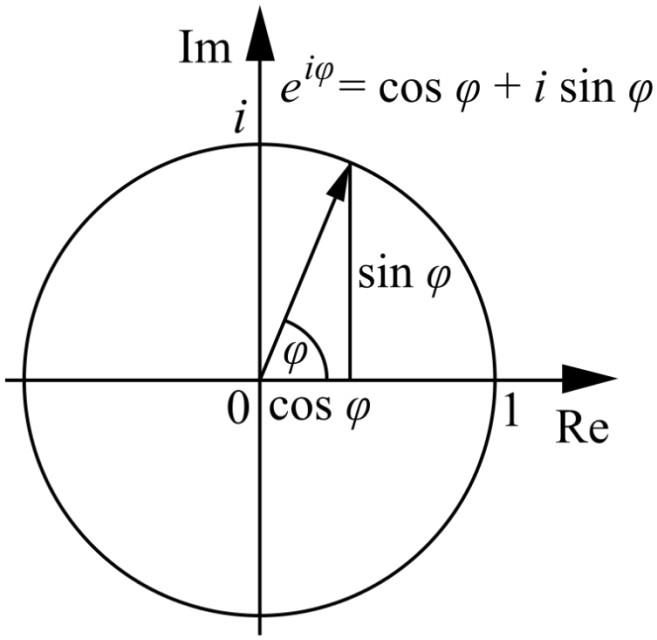


Source: Seul et al, *Practical Algorithms for Image Analysis*, 2000, p. 249, 262.

2D FFT can be computed as two discrete Fourier transforms in 1 dimension

Fast Fourier transform in python (OpenCV)

```
import cv2
img = cv2.imread(randomImage.jpg', cv2.IMREAD_GRAYSCALE) # load an image as grayscale

#Make sure the Image is in float32 precision (we're not displaying this image, we're only using it do some computations so no need to worry about img_as_float)  and compute the dft: this will return an array of size [width, height, 2], the first channel dft[:,:,0] is the Real part of the fourier transform and dft[:,:,1] is the imaginary part
dft = cv2.dft(np.float32(img), flags=cv2.DFT_COMPLEX_OUTPUT)
```



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

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

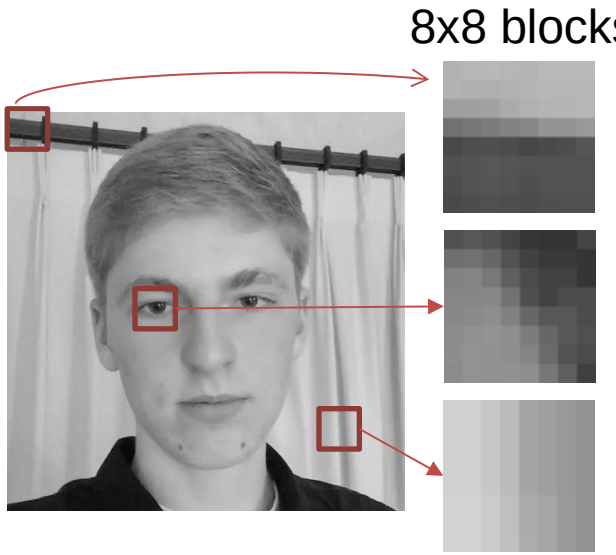
Fast Fourier transform in python (OpenCV)

```
#shift the transform so that it is centered in the middle. Use this to perform subsequent processing (filtering inverse fourtier transform etc.)
dft_shift = np.fft.fftshift(dft)

#For visualization purposed only:  compute and show the magnitude, the range of values is very large (typically from 0 to 1e6-> compute its log. However, log cannot handle 0, add 1 to the log to avoid errors), finally since the magnitude values may be small multiply them by a constant (20) so that they appear:
Magnitude_spect = 20* np.log(cv2.magnitude(dft_shift[:,:,0], dft_shift[:,:,1])+1)

plt.subplot(121),plt.imshow(img, cmap = 'gray')
plt.title('Input Image'), plt.xticks([]), plt.yticks([])
plt.subplot(122),plt.imshow(magnitude_spectrum, cmap = 'gray')
plt.title('Magnitude Spectrum'), plt.xticks([]),
plt.yticks([])
plt.show()
```

Thinking in Frequency - Compression

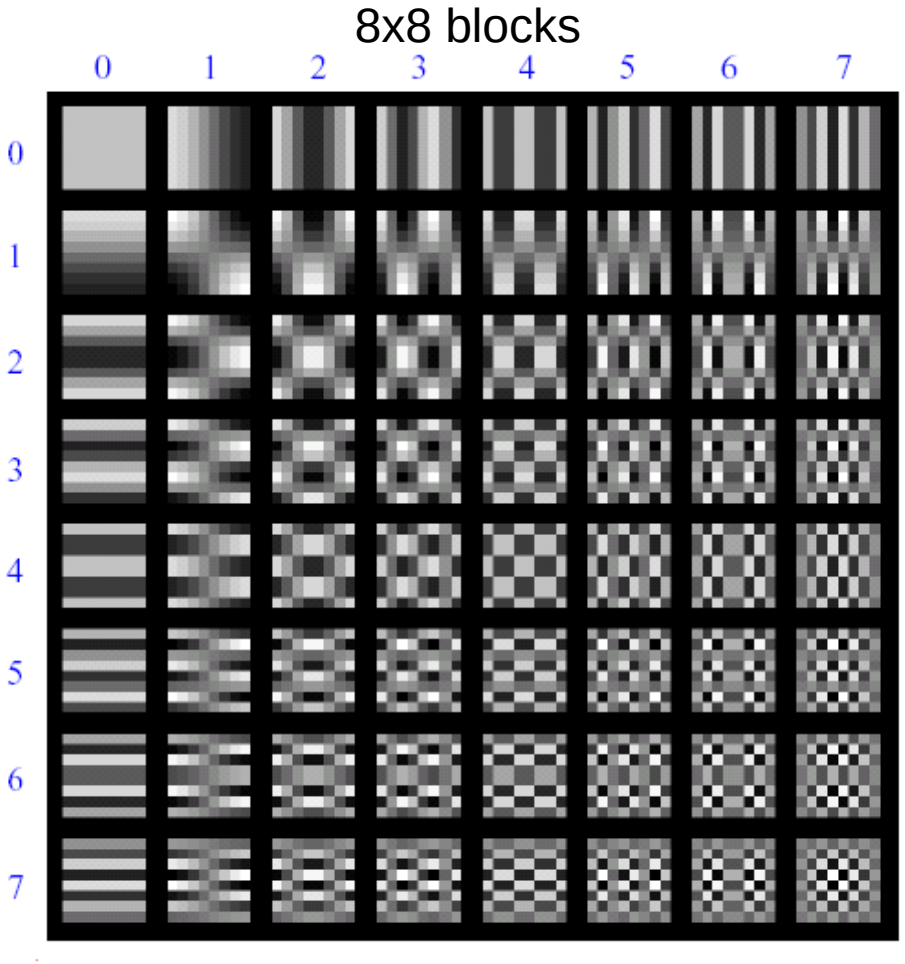
How is it that a 4MP image can be compressed to a few hundred KB without a noticeable change?

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)

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

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

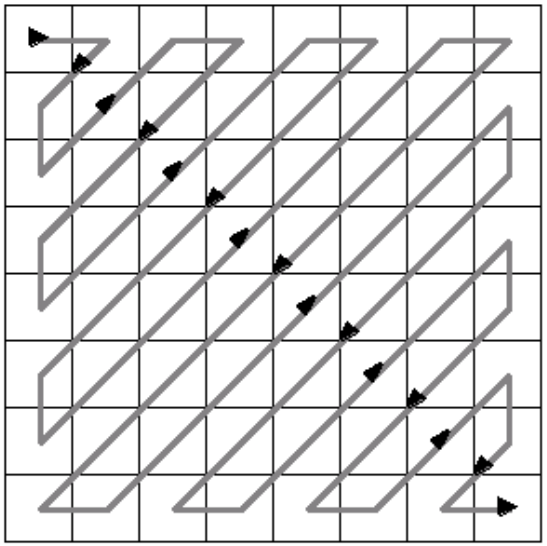
JPEG Encoding

- Entropy coding (Huffman-variant)

Quantized values

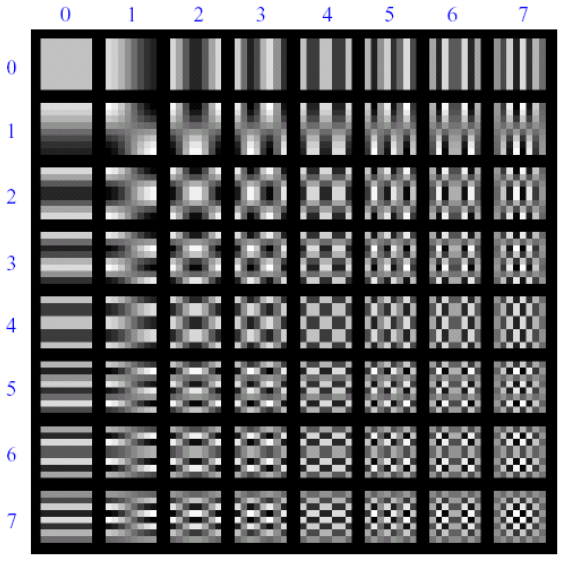
$$B = \begin{bmatrix} -26 & -3 & -6 & 2 & 2 & -1 & 0 & 0 \\ 0 & -2 & -4 & 1 & 1 & 0 & 0 & 0 \\ -3 & 1 & 5 & -1 & -1 & 0 & 0 & 0 \\ -3 & 1 & 2 & -1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Linearize *B* like this.



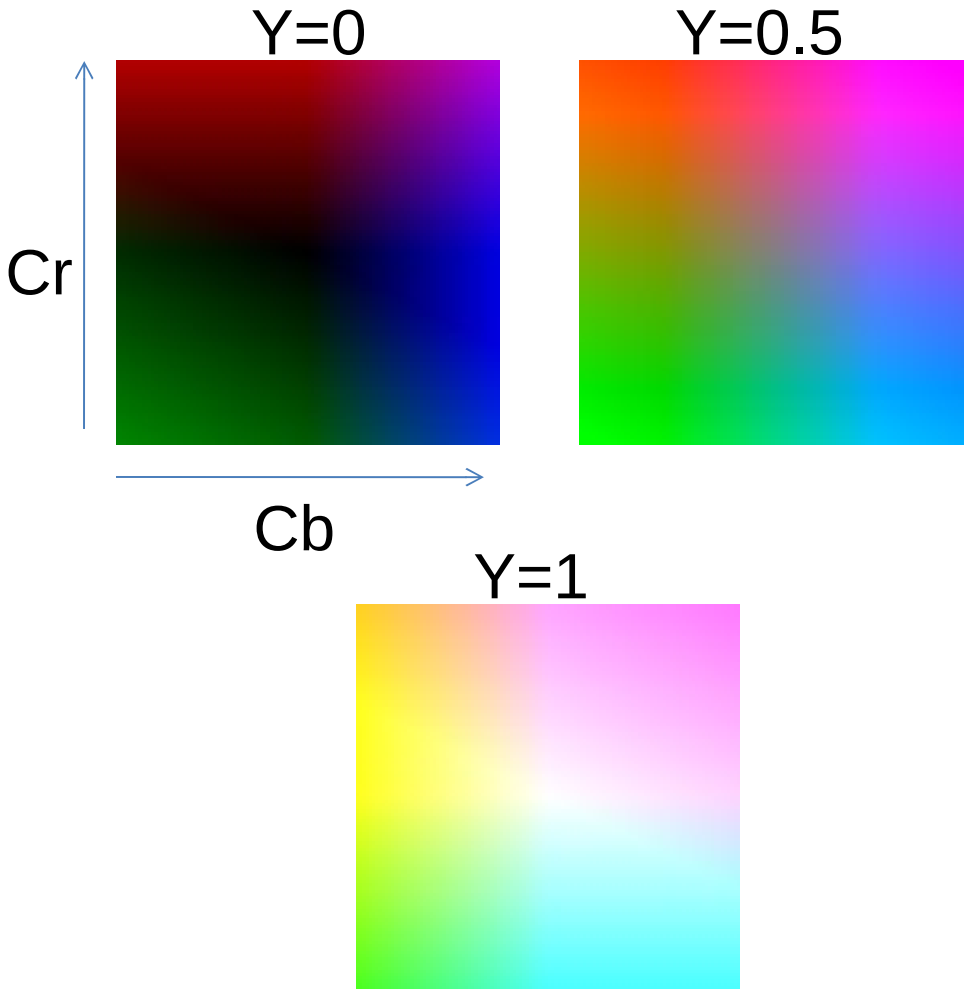
Helps compression:

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



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