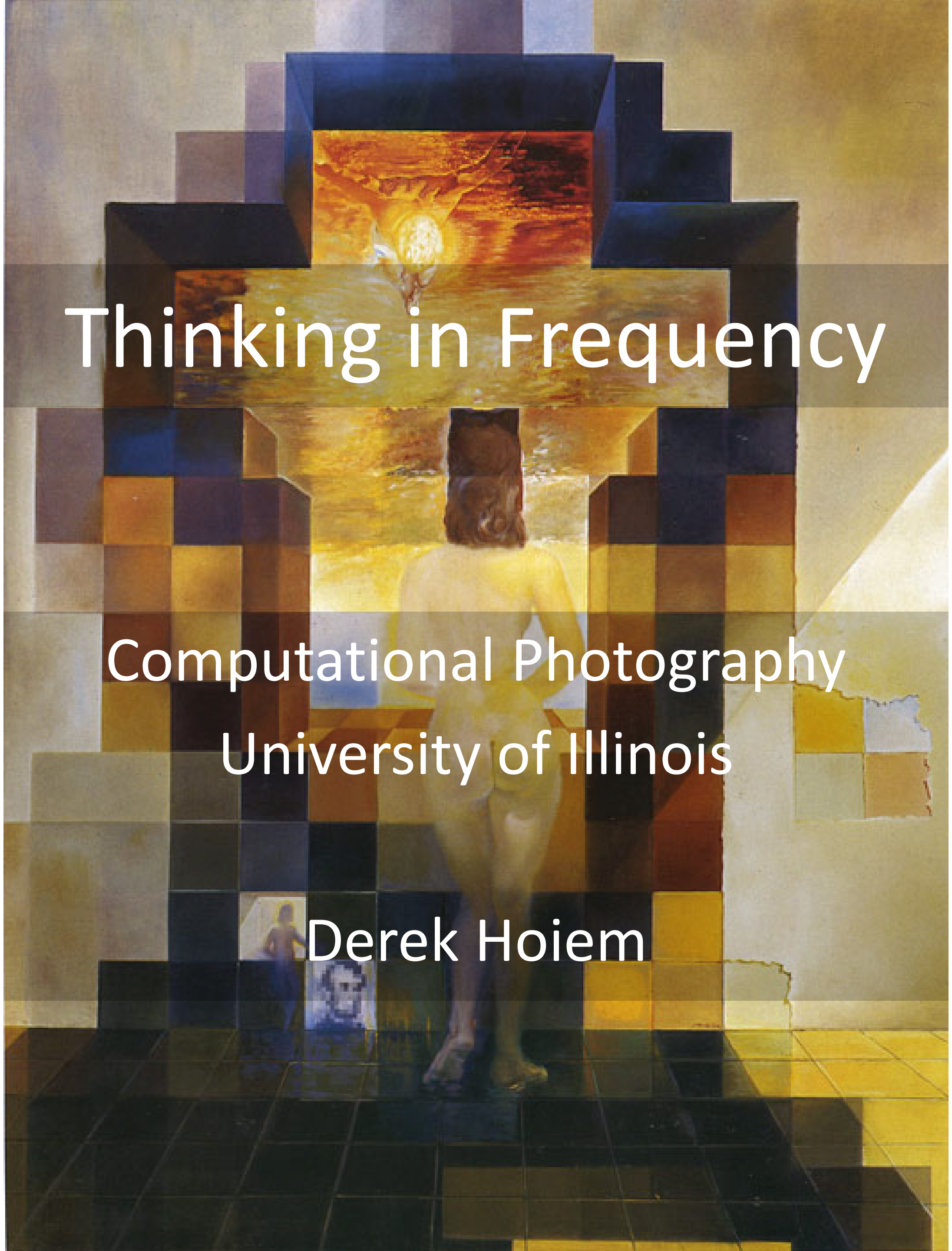
Thinking in Frequency

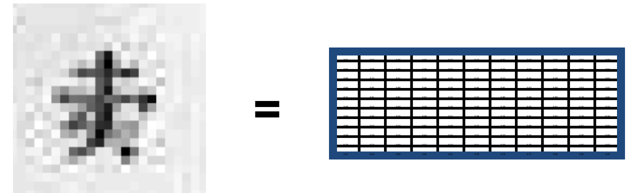


Computational Photography
University of Illinois

Derek Hoiem

Last class

- Image is a matrix of numbers



- Linear filtering is a dot product at each position
  - Can smooth, sharpen, translate (among many other uses)

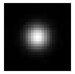
$$\frac{1}{9}\begin{bmatrix}1 & 1 & 1\\ 1 & 1 & 1\\ 1 & 1 & 1\end{bmatrix}$$

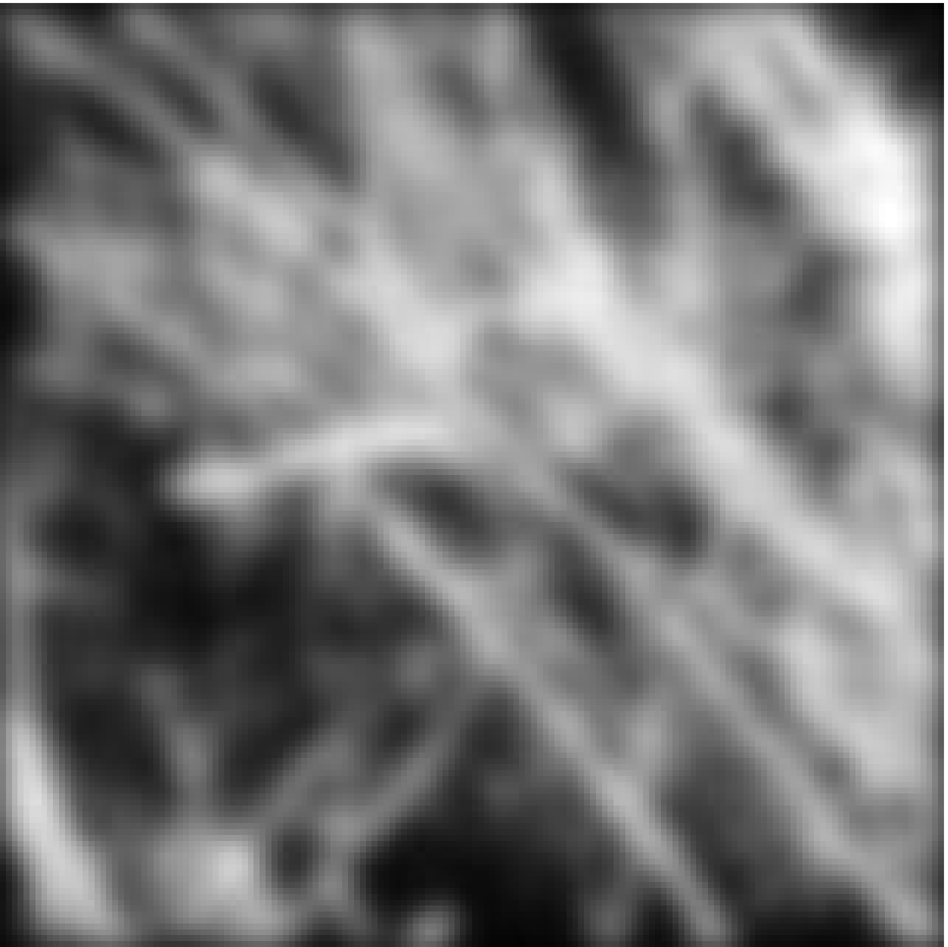
Today’s class

- Fourier transform and frequency domain
  - Frequency view of filtering
  - Another look at hybrid images
  - Sampling

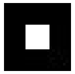
Why does the Gaussian give a nice smooth image, but the square filter give edgy artifacts?

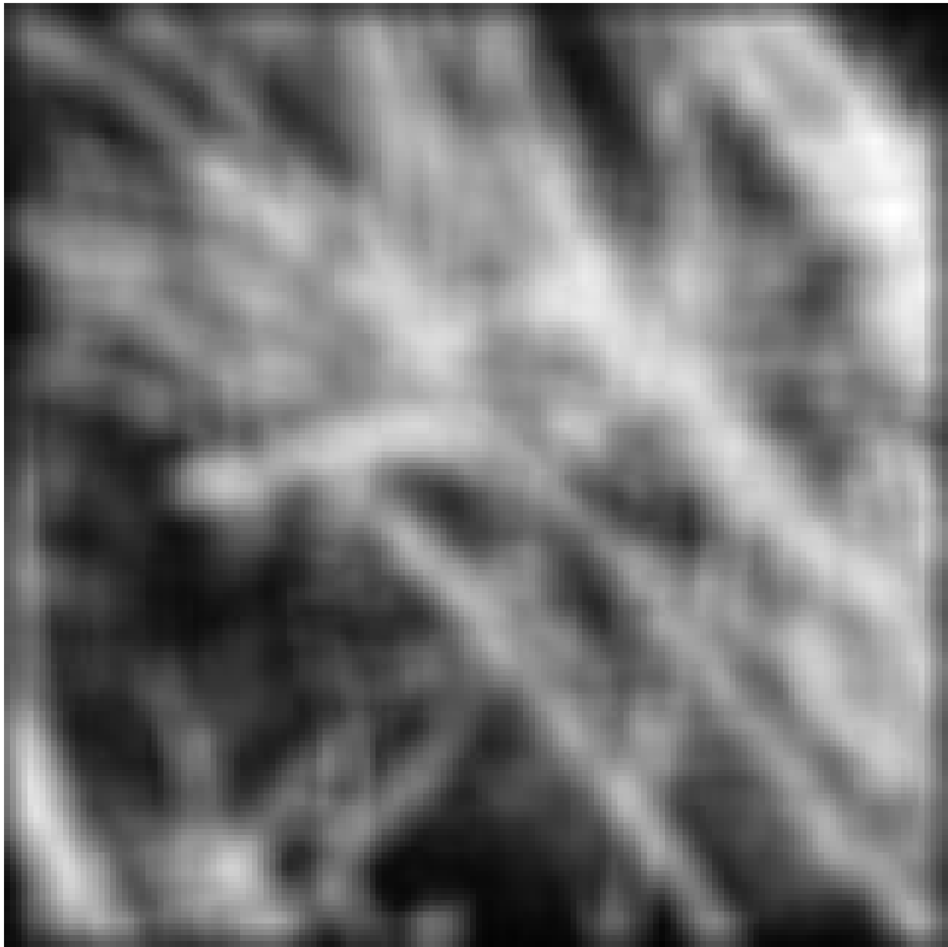
Gaussian



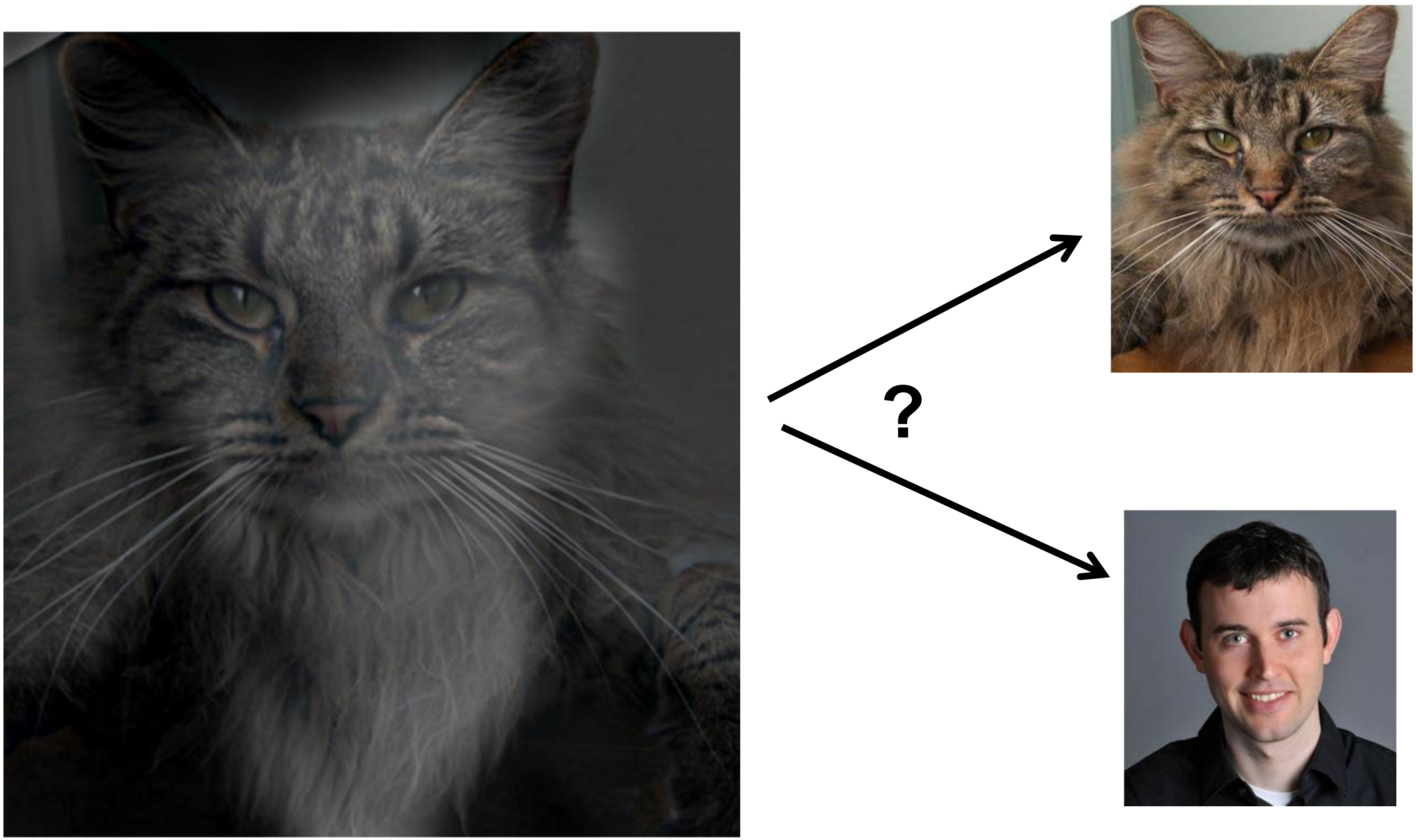


Box filter





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



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

Image: http://www.flickr.com/photos/igorms/136916757/

Thinking in terms of frequency

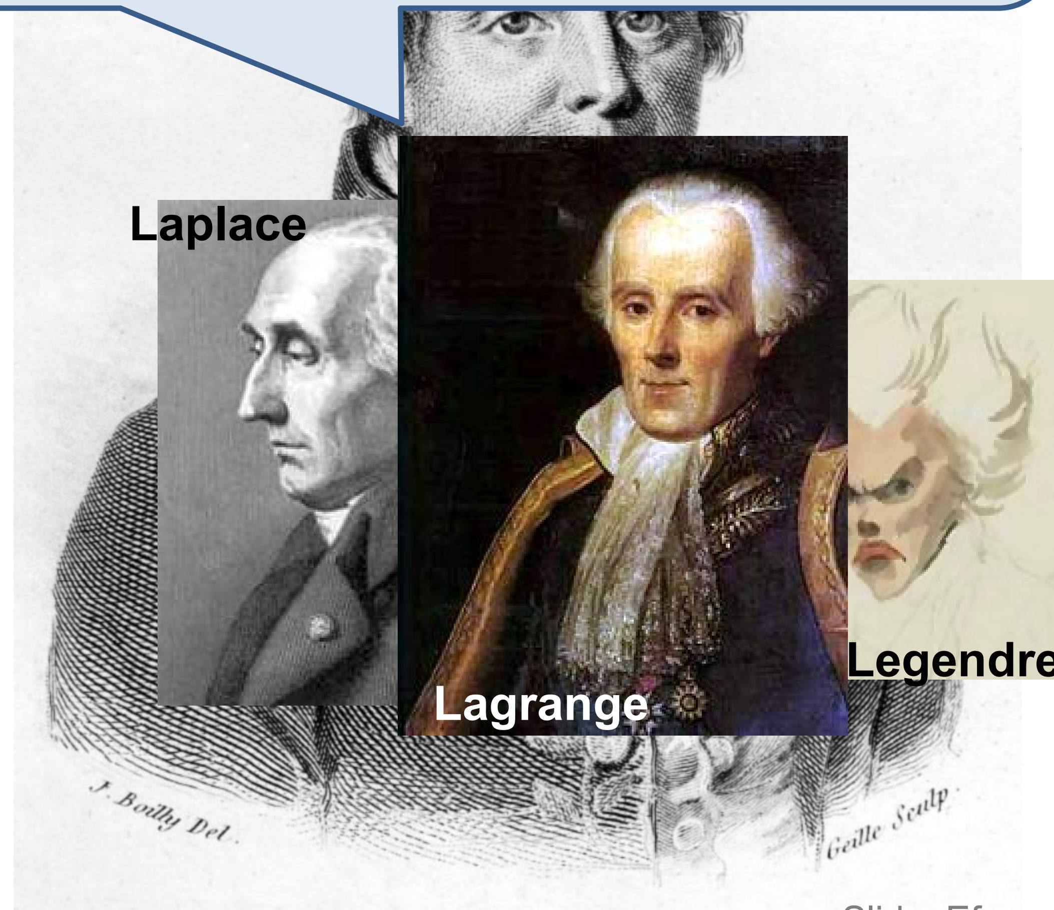
Jean Baptiste Joseph Fourier (1768-1830)

had crazy idea (1807):

> *...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.*

***Any** univariate function can*
rewritten as a weighted sum
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
  - there are some subtle restrictions



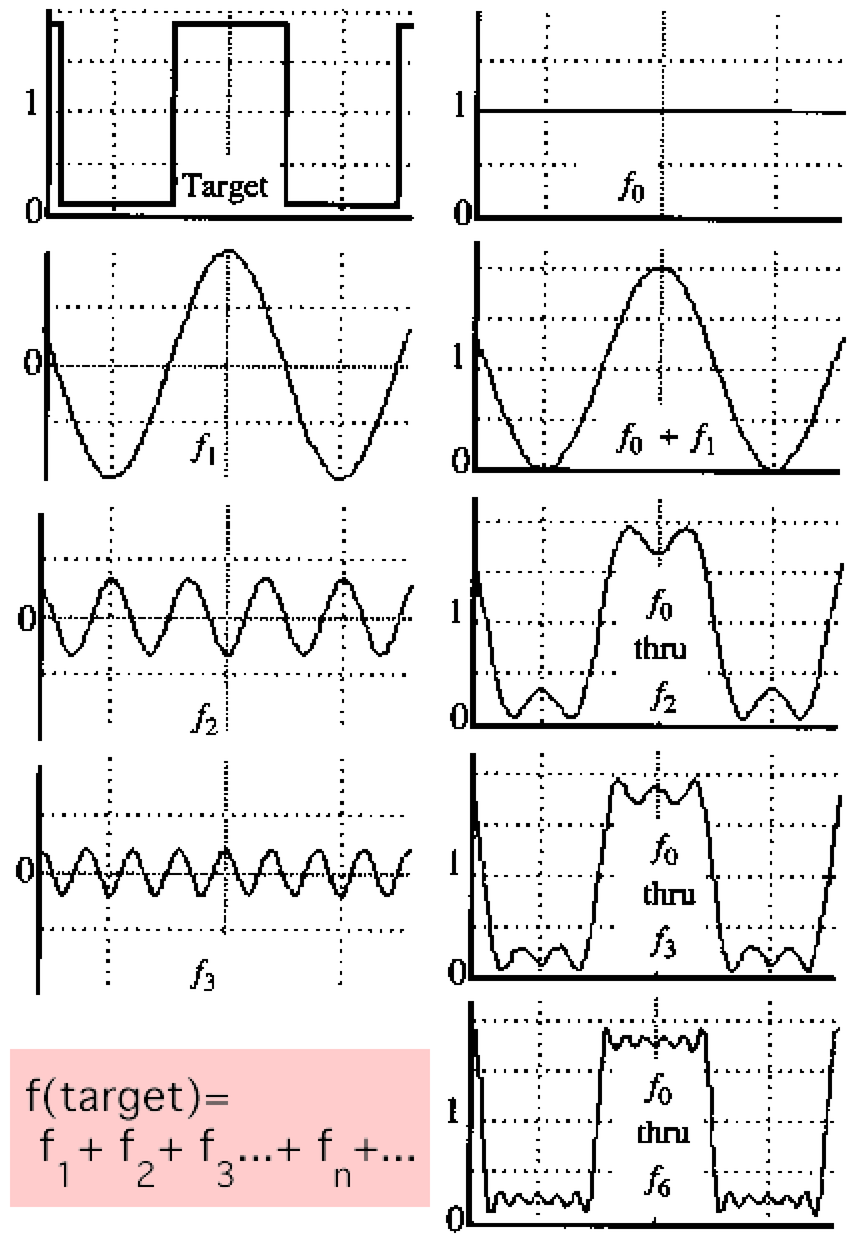
Slide: Efros

A sum of sines

Our building block:

$$A\sin(\omega x + \phi)$$

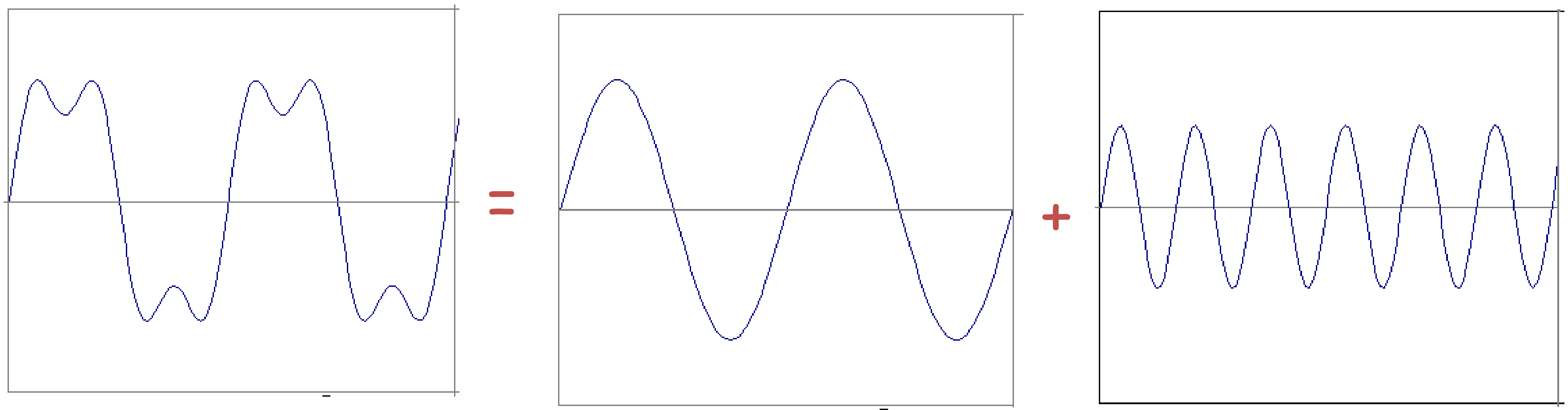
Add enough of them to get any signal *f(x)* you want!

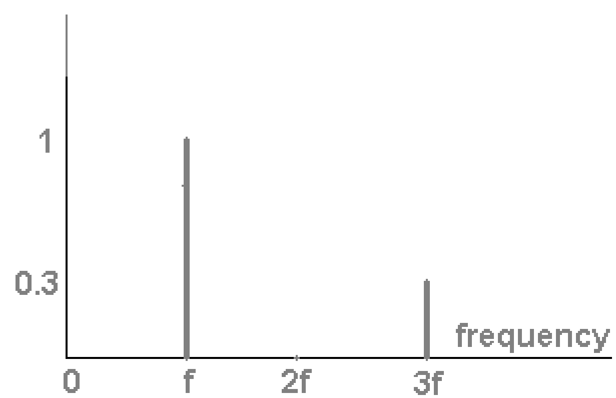


f(target)=
$f_1 + f_2 + f_3 ... + f_n + ...$

Frequency Spectra

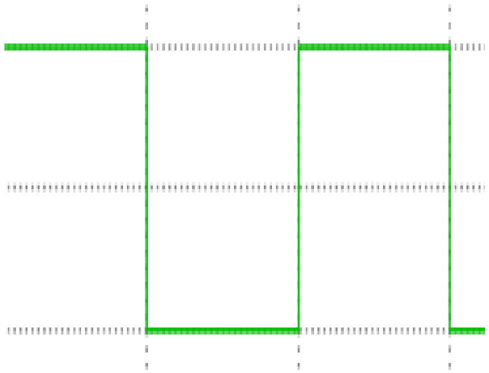
- example : $g(t) = \sin(2\pi f\, t) + (1/3)\sin(2\pi(3f)\, t)$



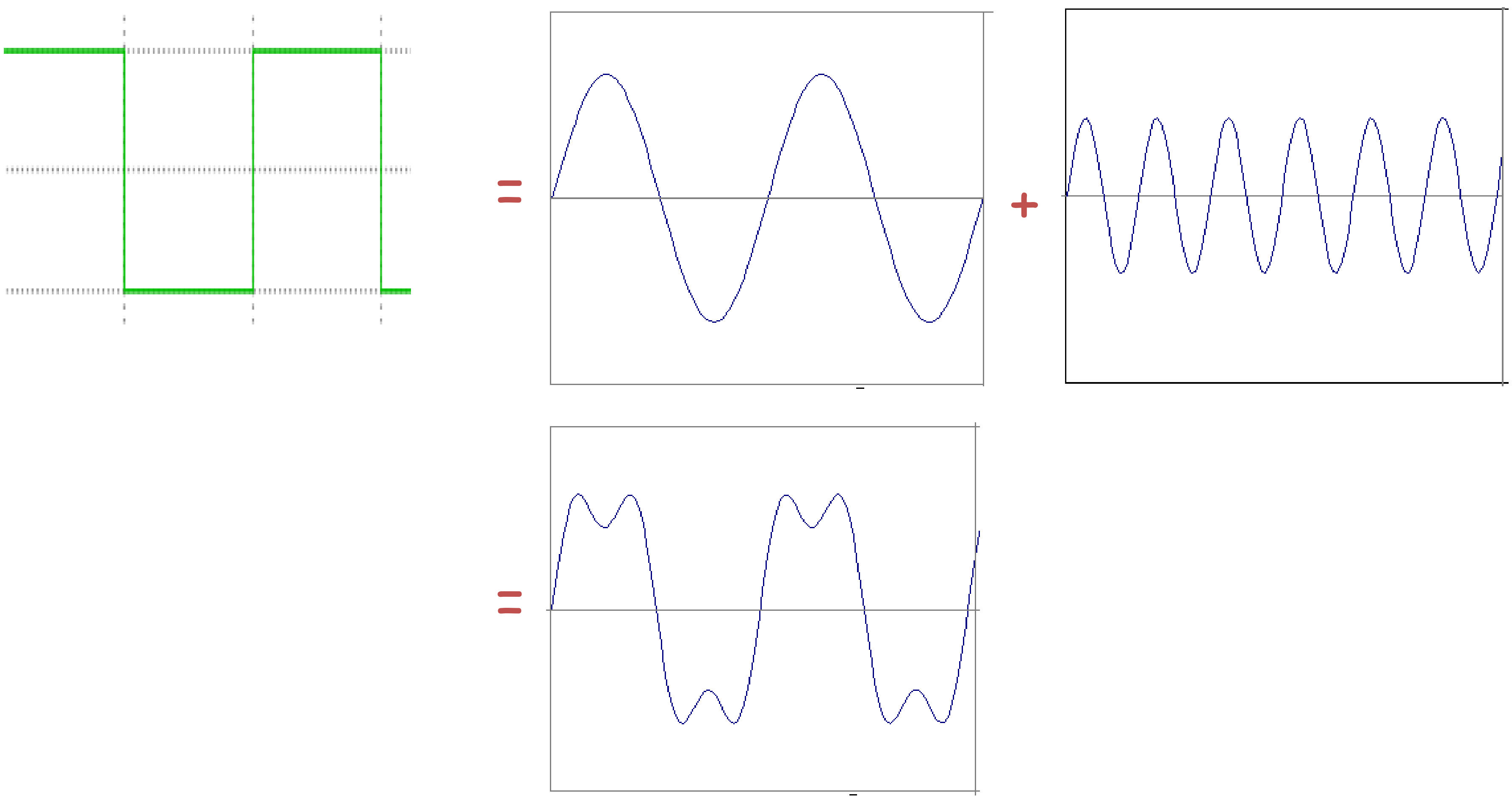


Slides: Efros

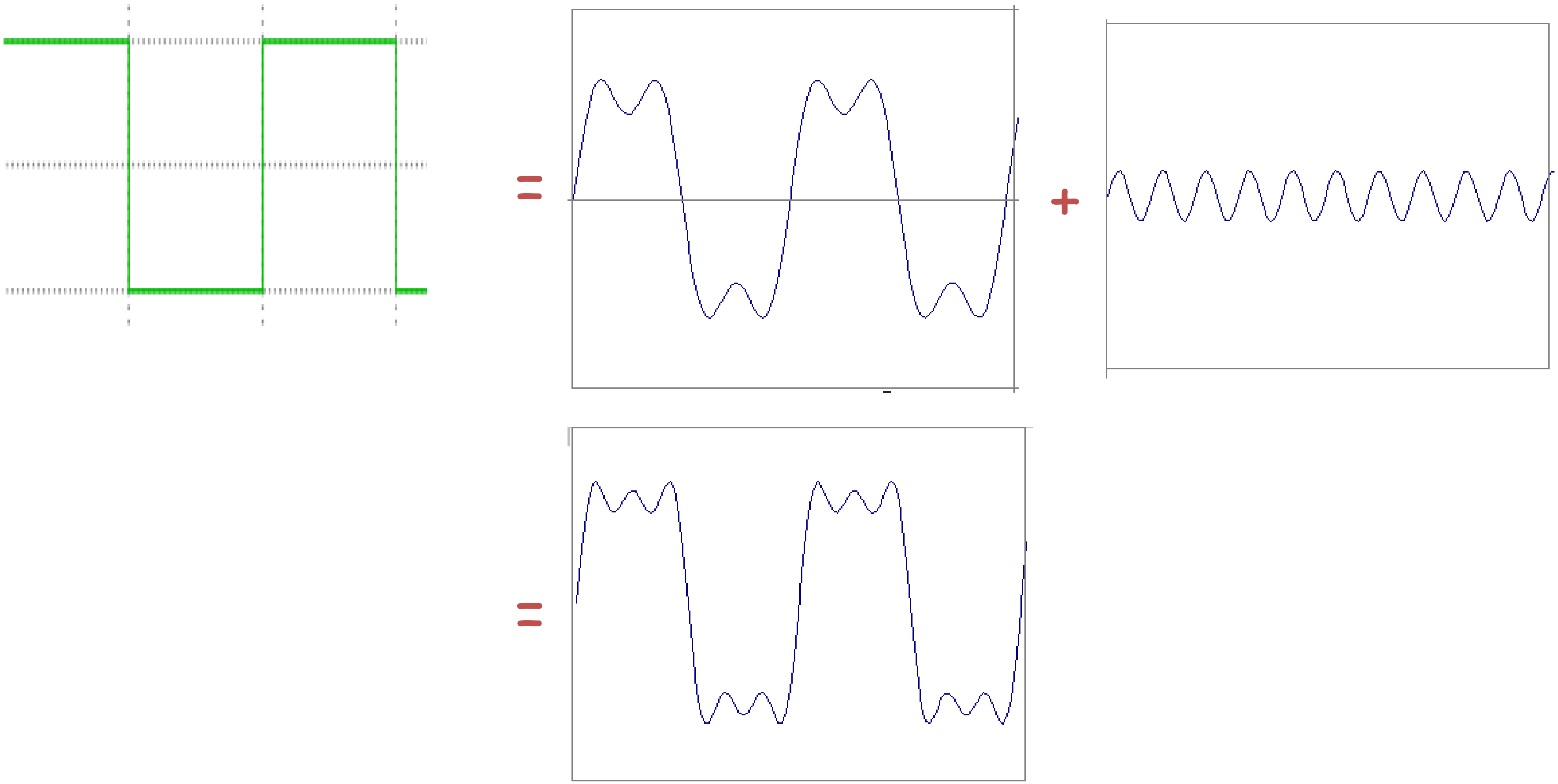
Frequency Spectra



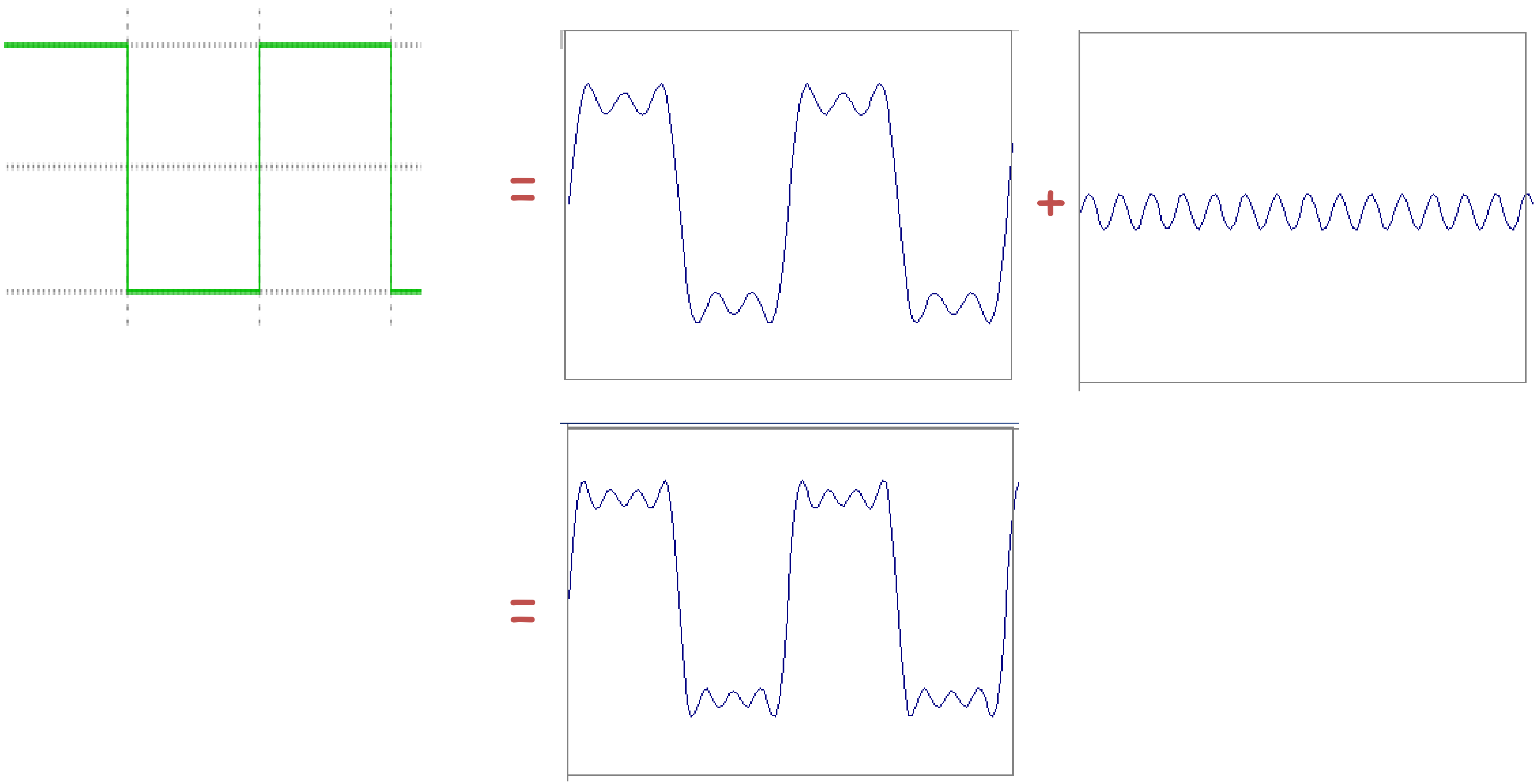
Frequency Spectra



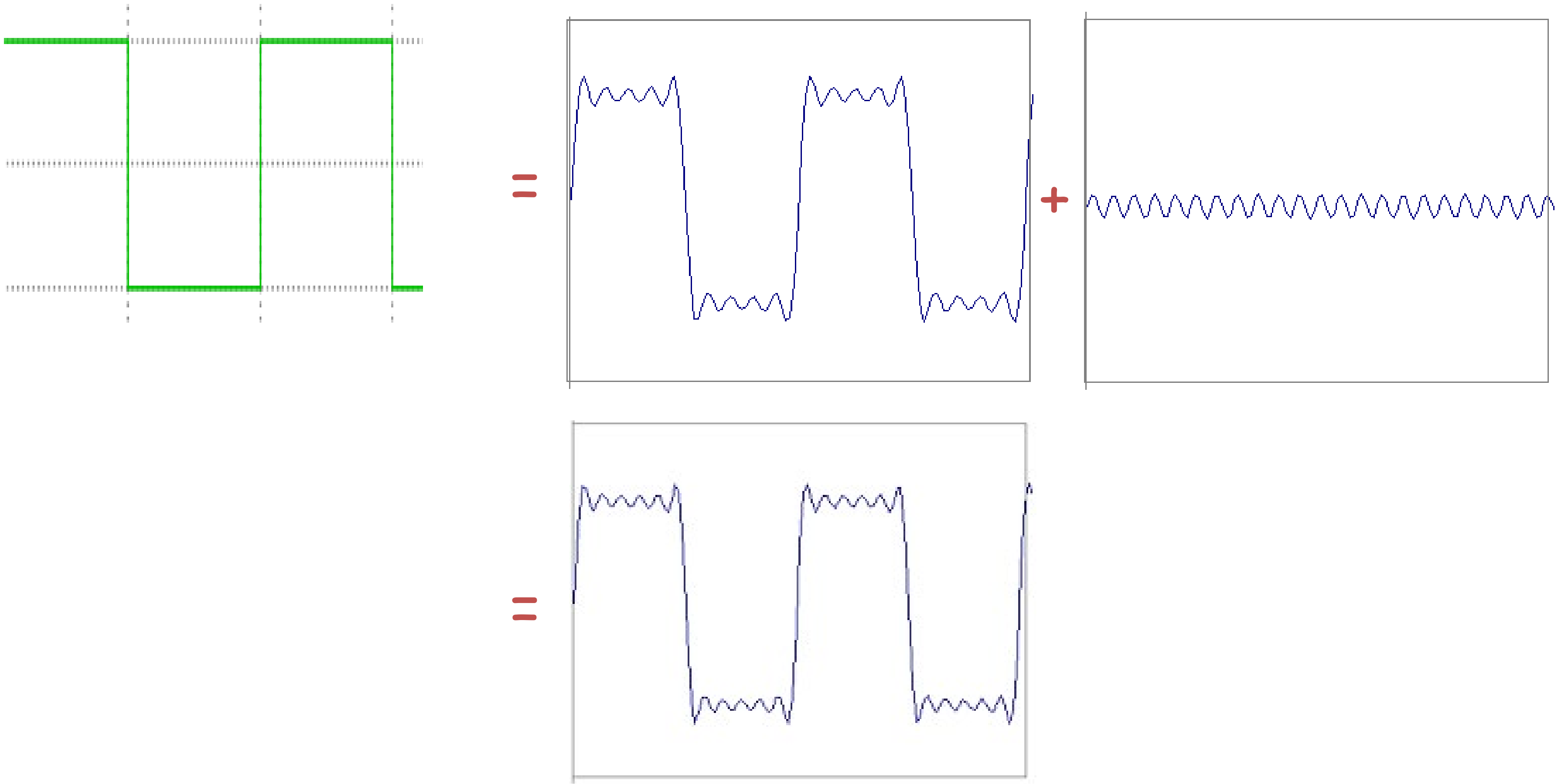
Frequency Spectra



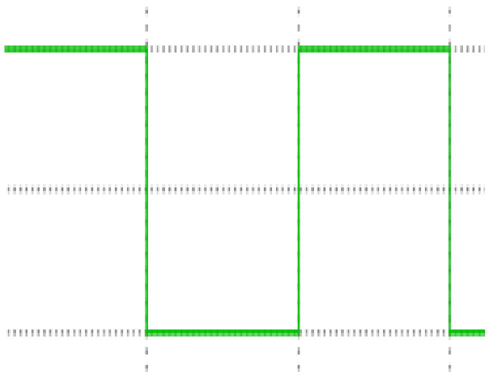
Frequency Spectra



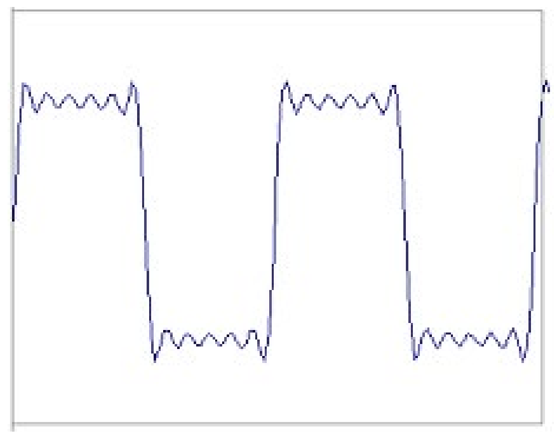
Frequency Spectra



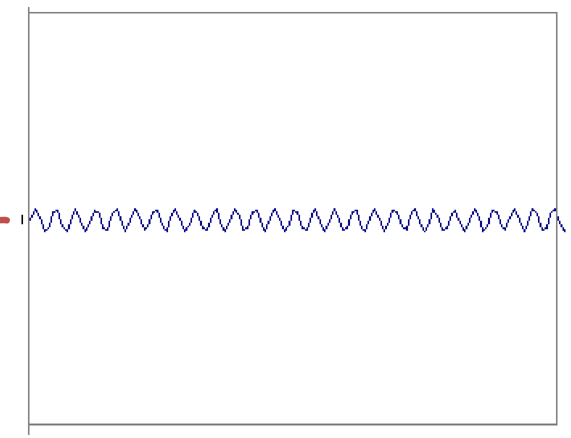
Frequency Spectra



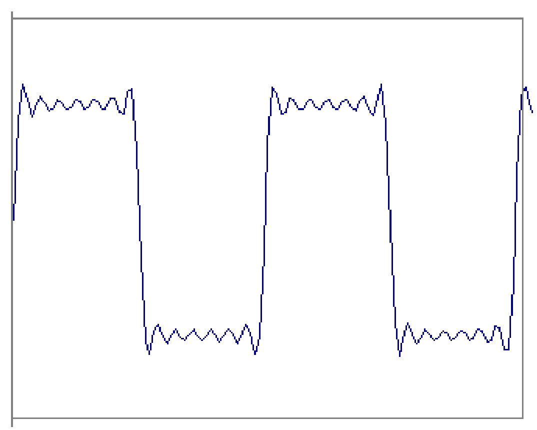
=



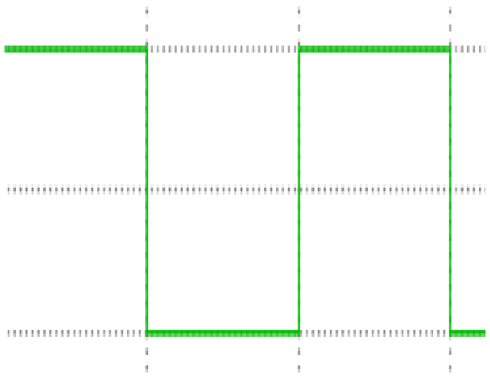
+



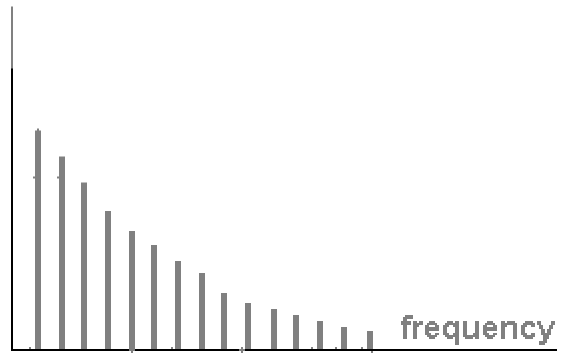
=



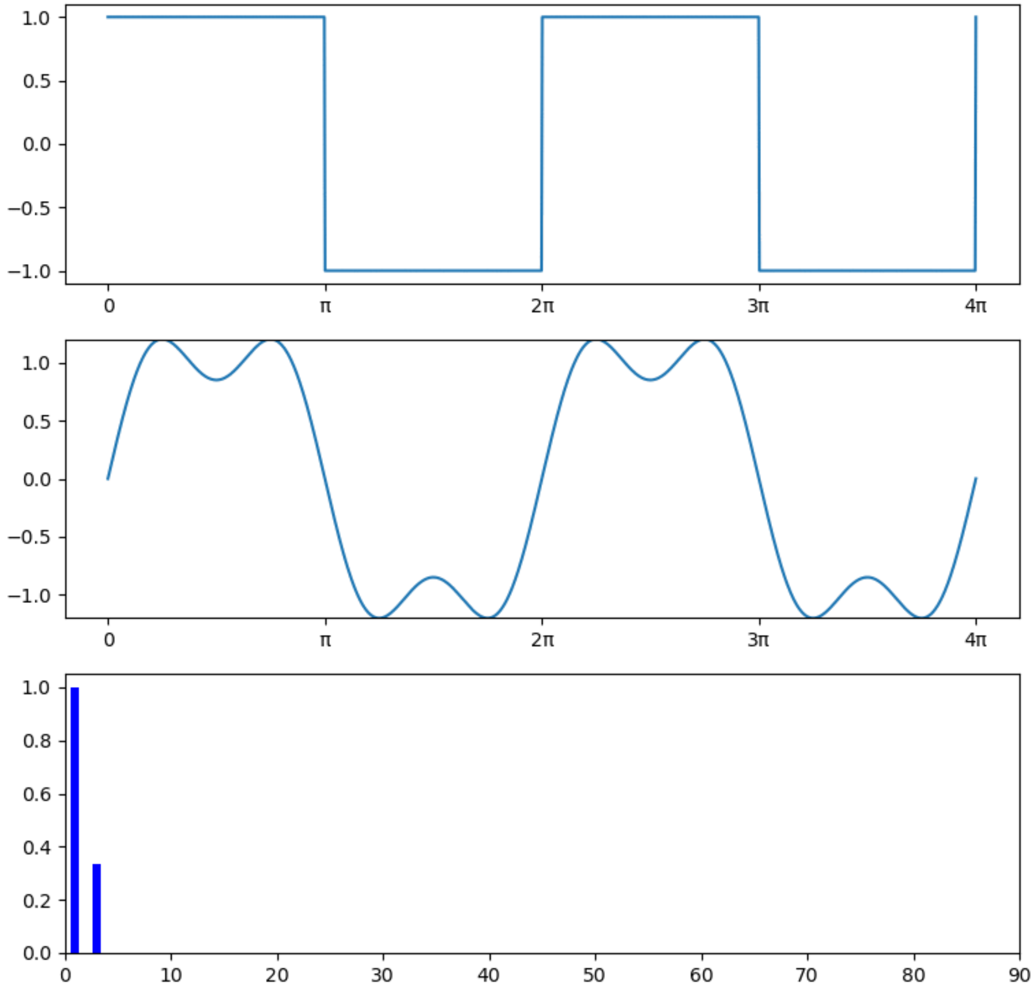
Frequency Spectra



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

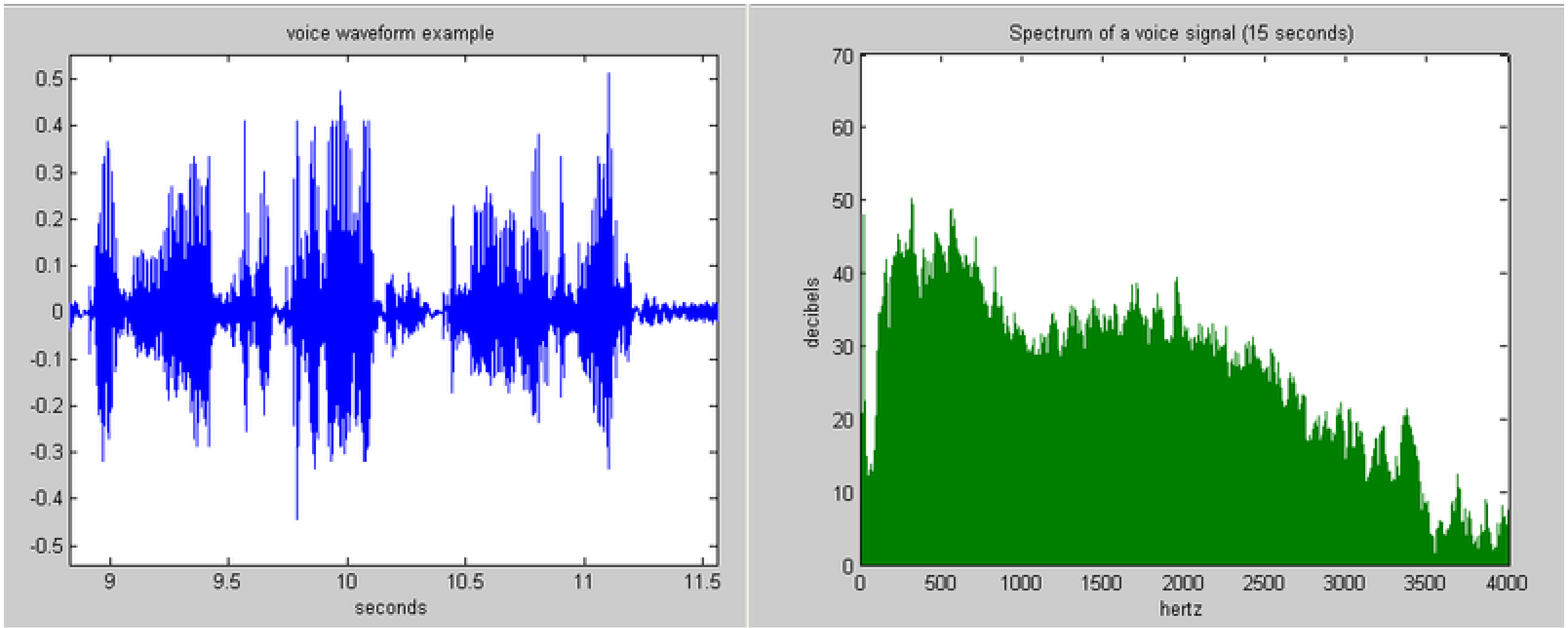


Frequency Spectra



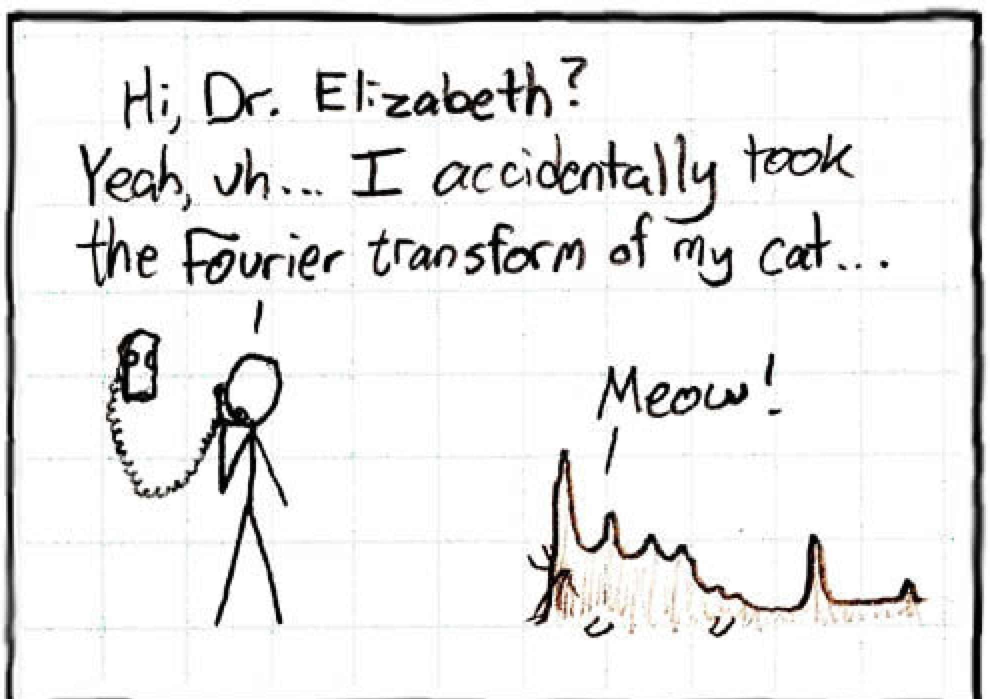
Example: Music

- We think of music in terms of frequencies at different magnitudes



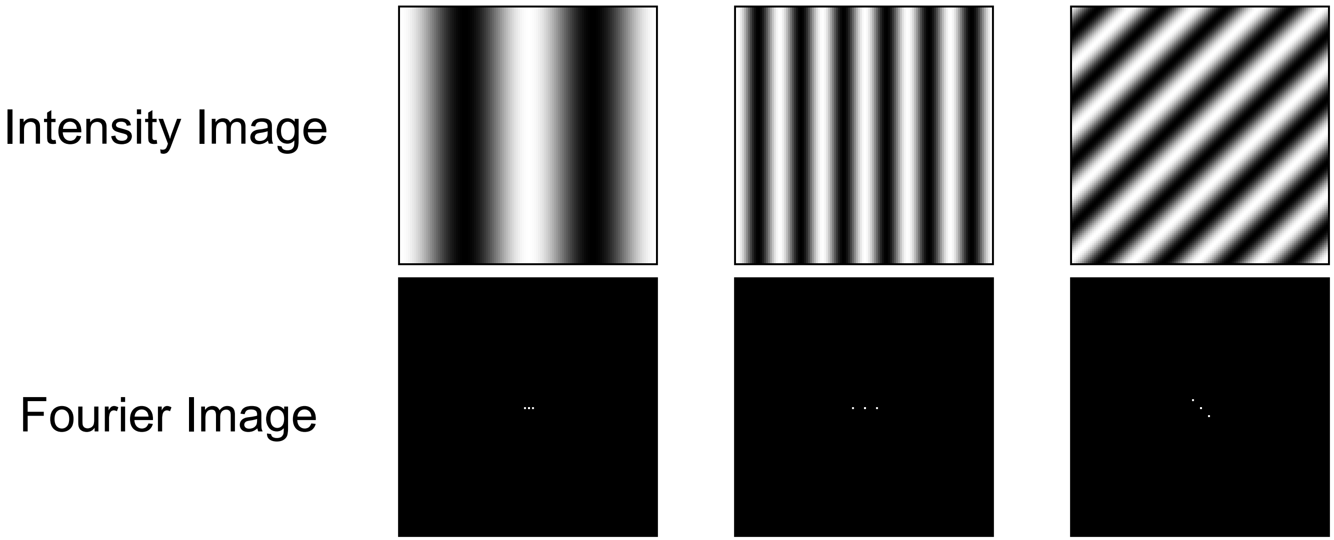
Other signals

- We can also think of all kinds of other signals the same way



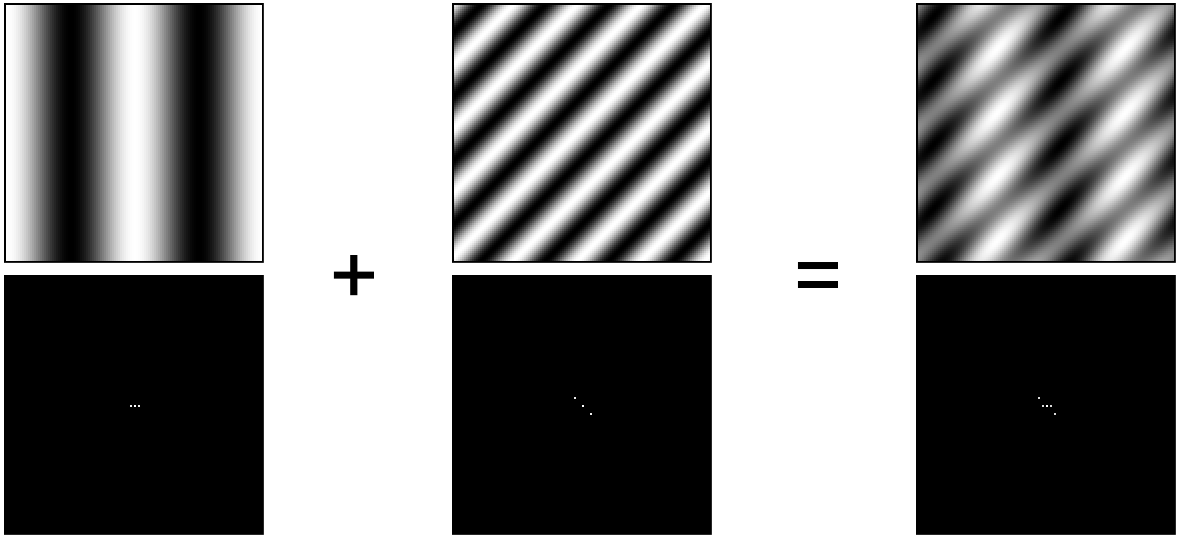
xkcd.com

Fourier analysis in images



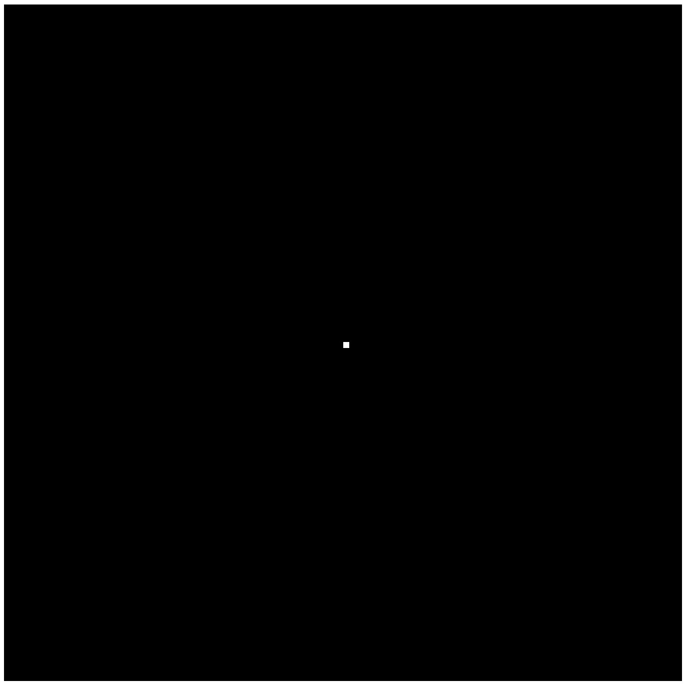
http://sharp.bu.edu/~slehar/fourier/fourier.html#filtering

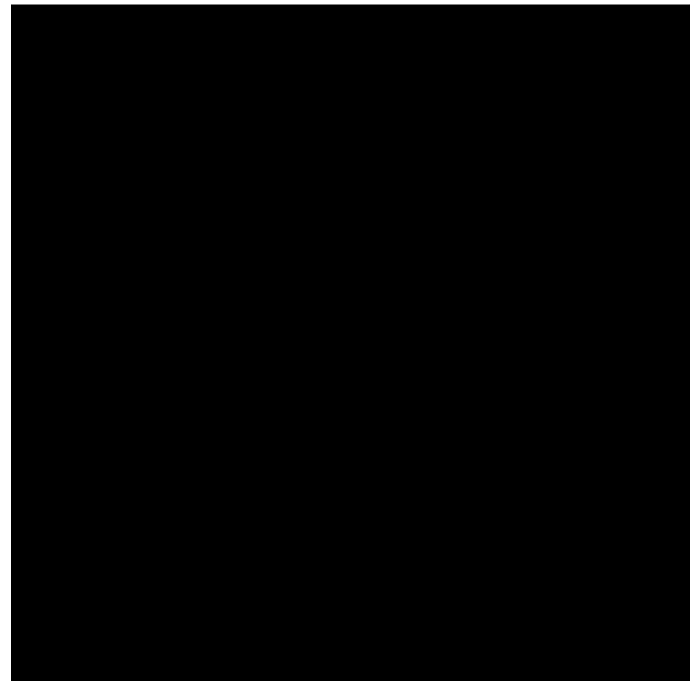
Signals can be composed



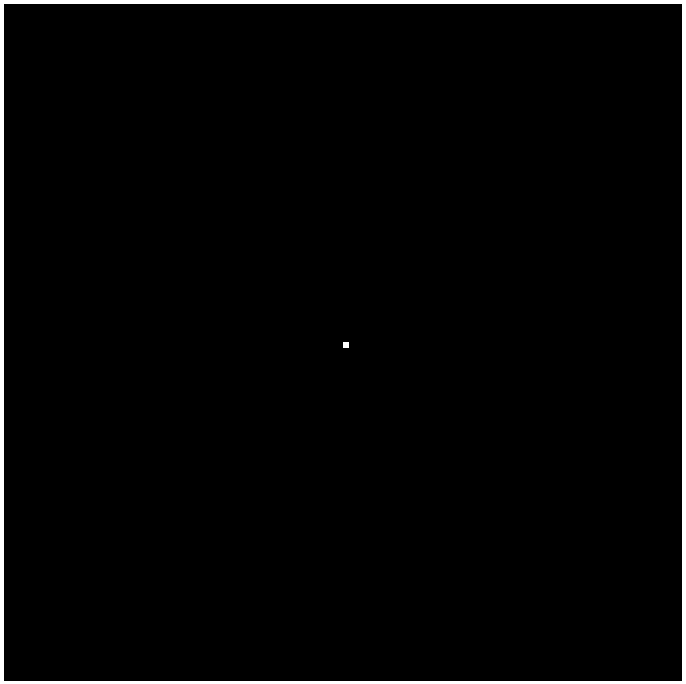
http://sharp.bu.edu/~slehar/fourier/fourier.html#filtering
More: http://www.cs.unm.edu/~brayer/vision/fourier.html

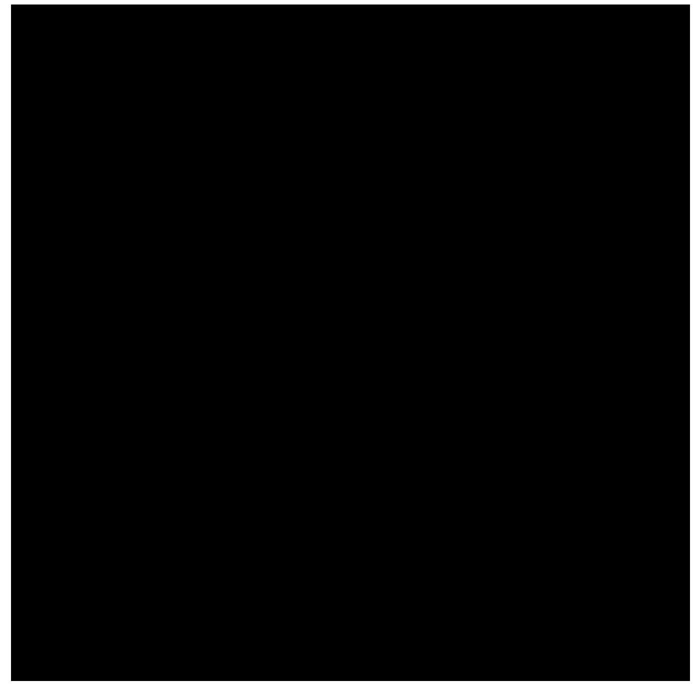
How change in frequency affects signal



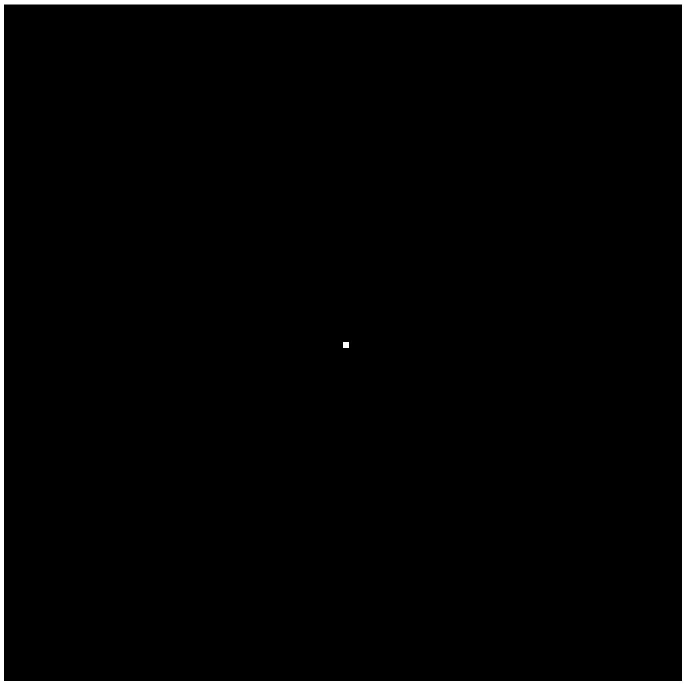


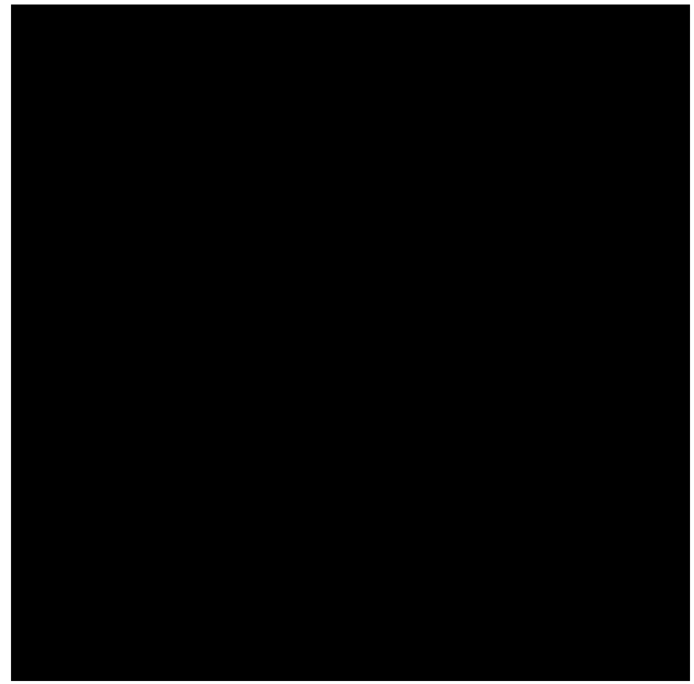
How change in frequency affects signal



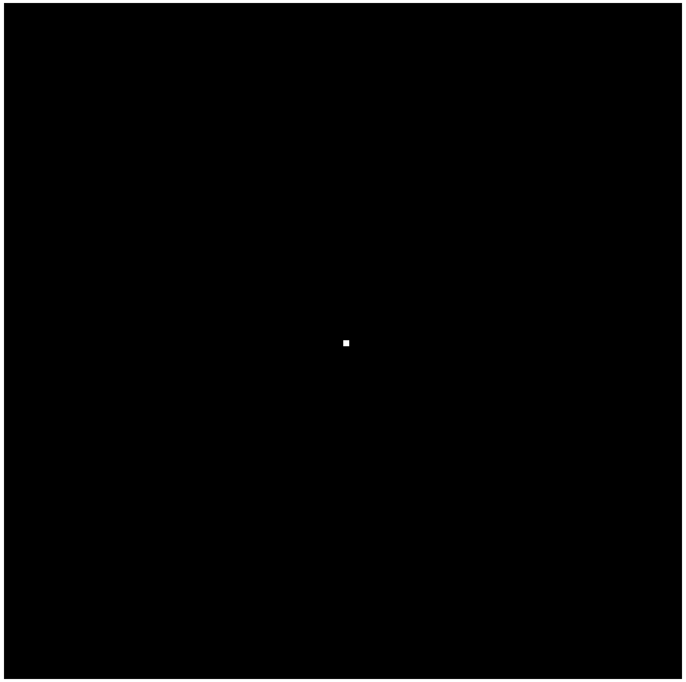


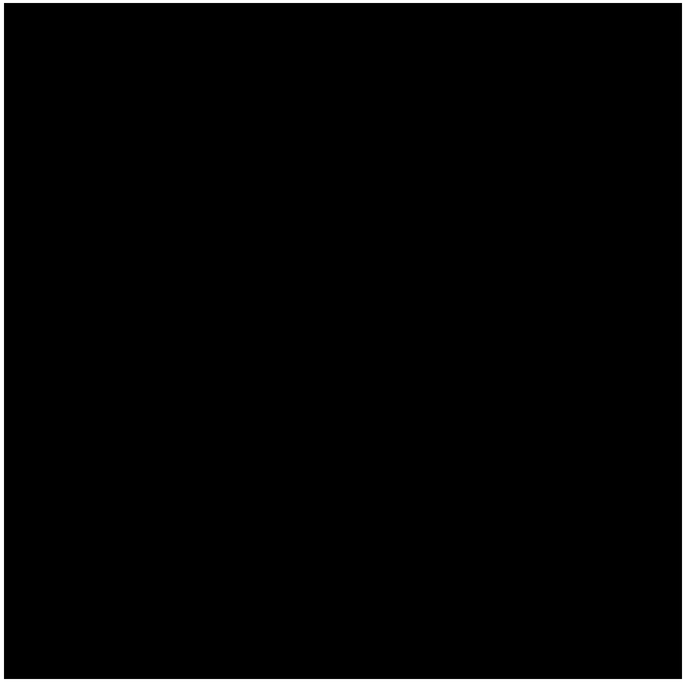
How change in frequency affects signal





How change in frequency affects signal





Fourier Transform

- Fourier transform stores the magnitude and phase at each frequency
  - Magnitude encodes how much signal there is at a particular frequency
  - Phase encodes spatial information (indirectly)
  - For mathematical convenience, this is often notated in terms of complex numbers

Amplitude: $A = \pm\sqrt{R(\omega)^2 + I(\omega)^2}$

Phase: $\phi = \tan^{-1} \frac{I(\omega)}{R(\omega)}$

Euler's formula: $e^{inx} = \cos(nx) + i\sin(nx)$

Computing the Fourier Transform

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

Continuous

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

Discrete

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

k=-N/2..N/2

Fast Fourier Transform (FFT): NlogN

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!

Properties of Fourier Transforms

- Linearity $\mathcal{F}[ax(t) + by(t)] = a\mathcal{F}[x(t)] + b\mathcal{F}[y(t)]$

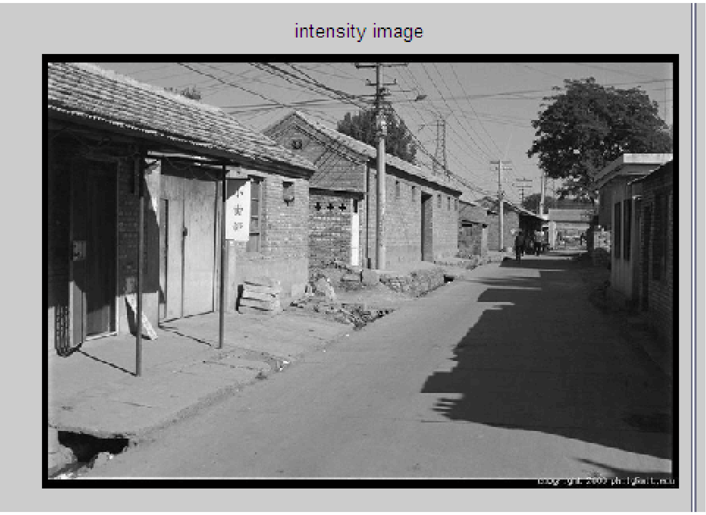
- Fourier transform of a real signal is symmetric about the origin

- The energy of the signal is the same as the energy of its Fourier transform

See Szeliski Book (3.4)

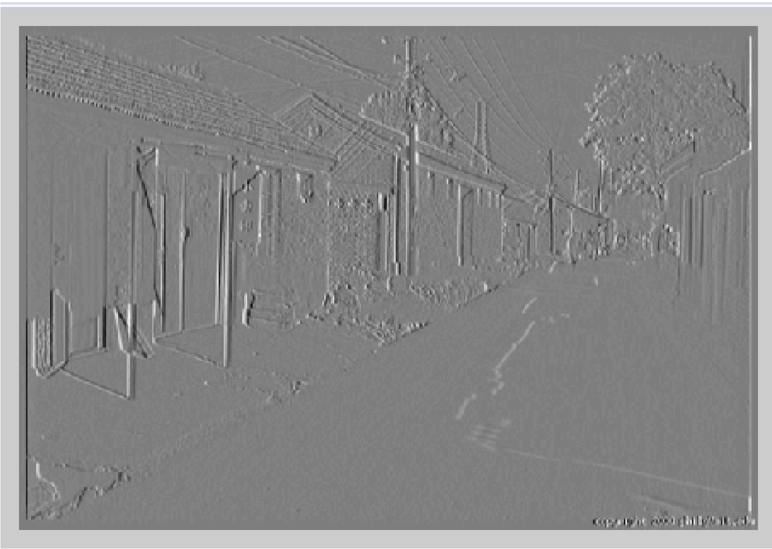
Filtering in spatial domain

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

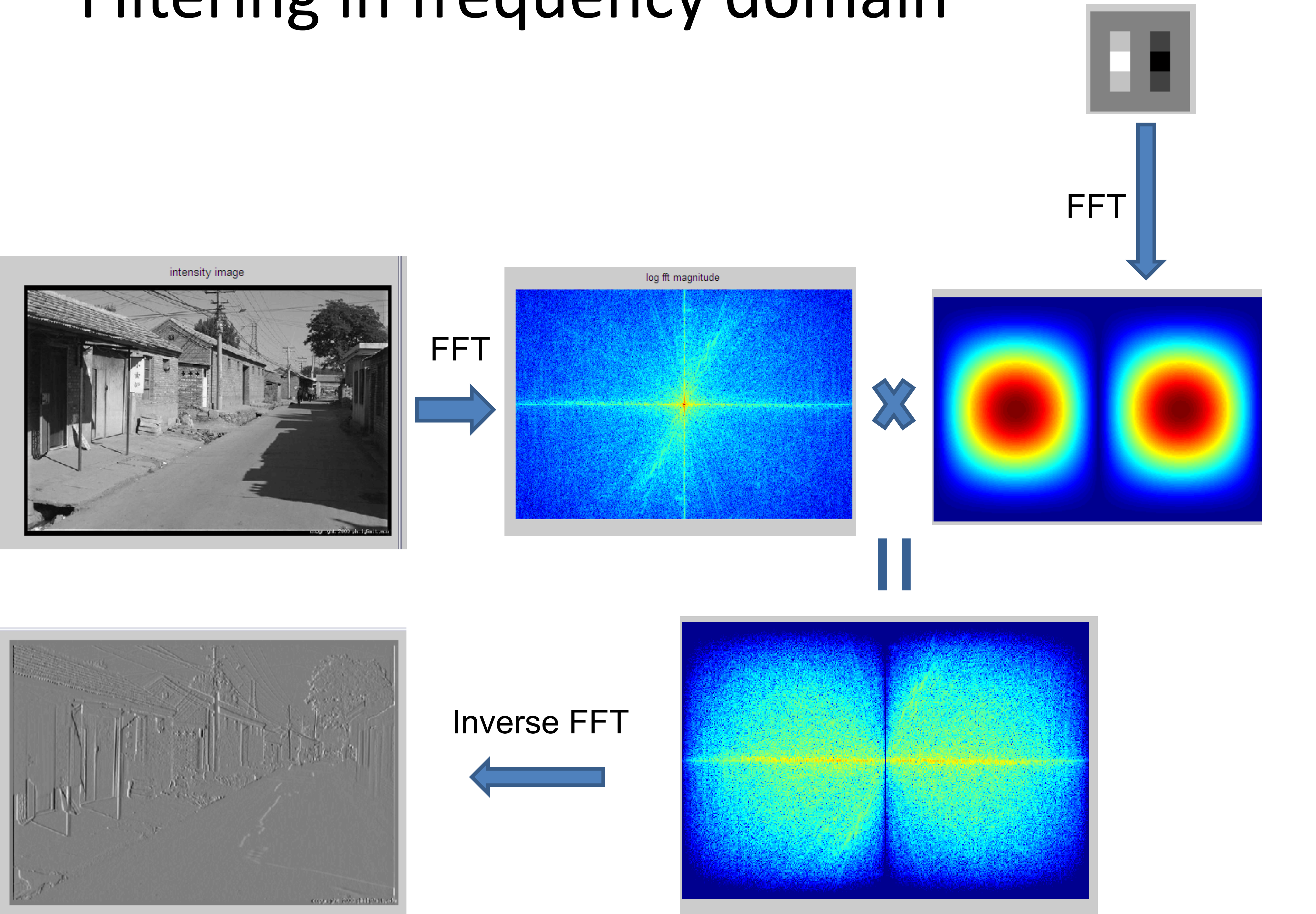


*

=



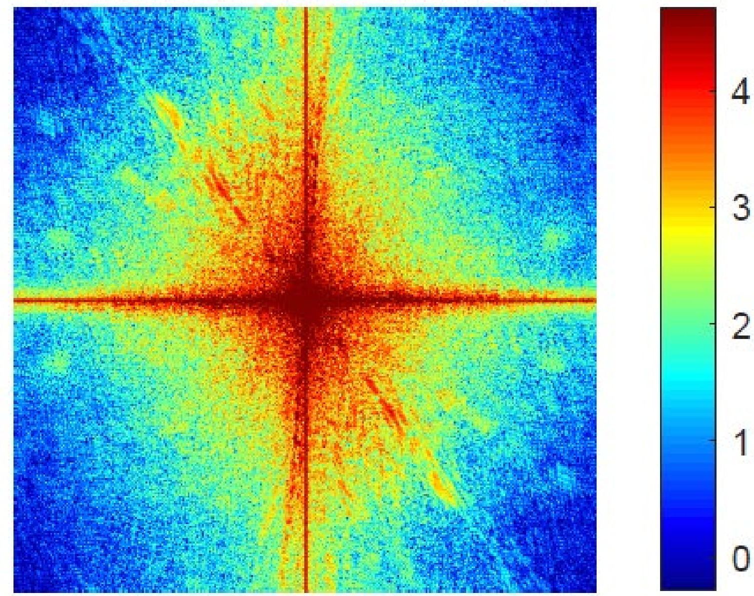
Filtering in frequency domain



Fourier Image Examples

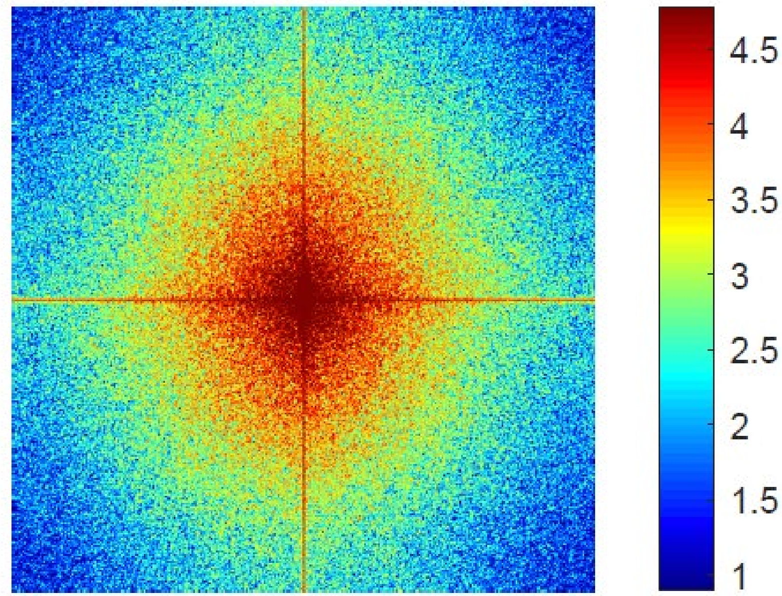
intensity image

log fft magnitude



intensity image

log fft magnitude

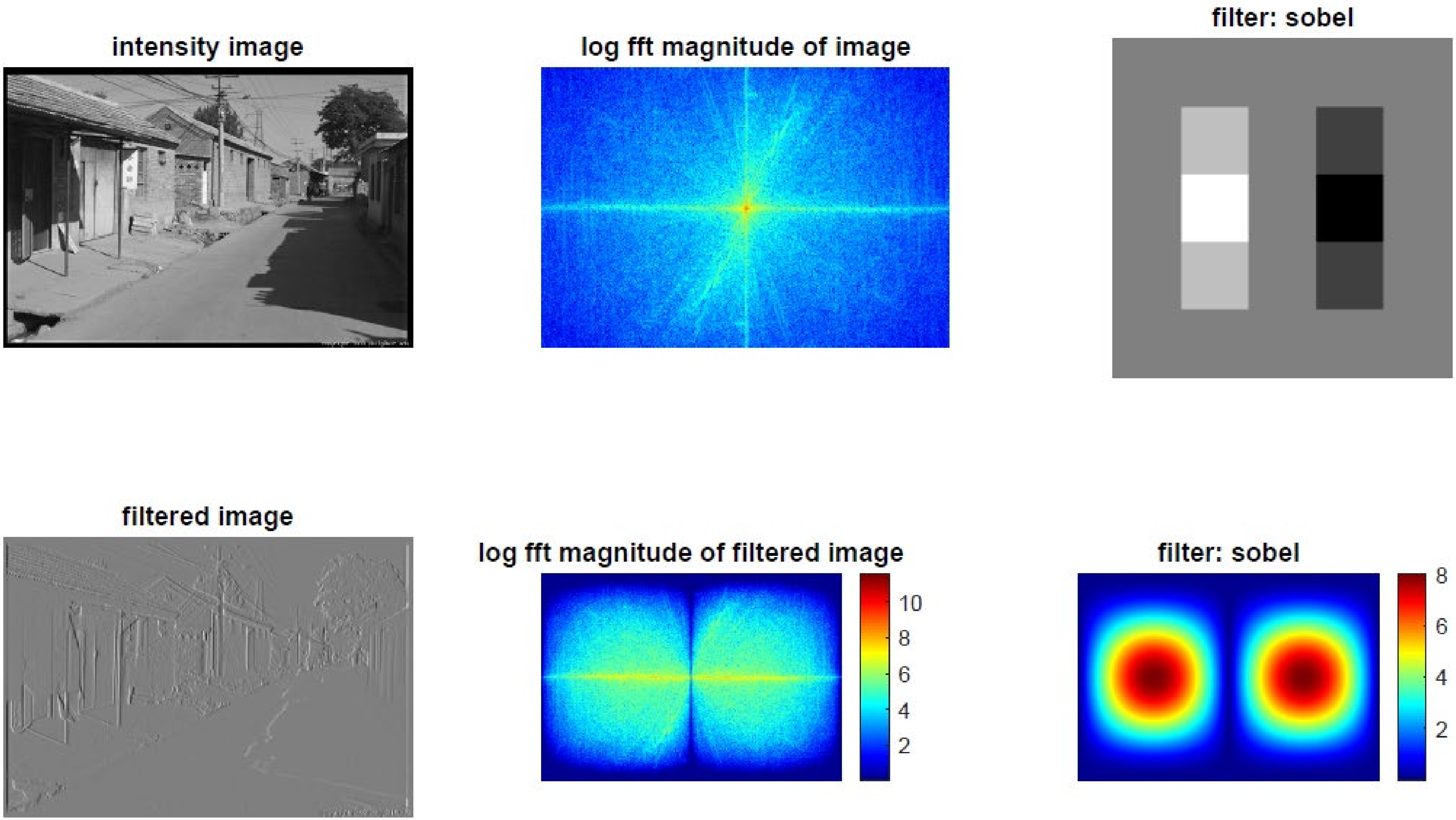


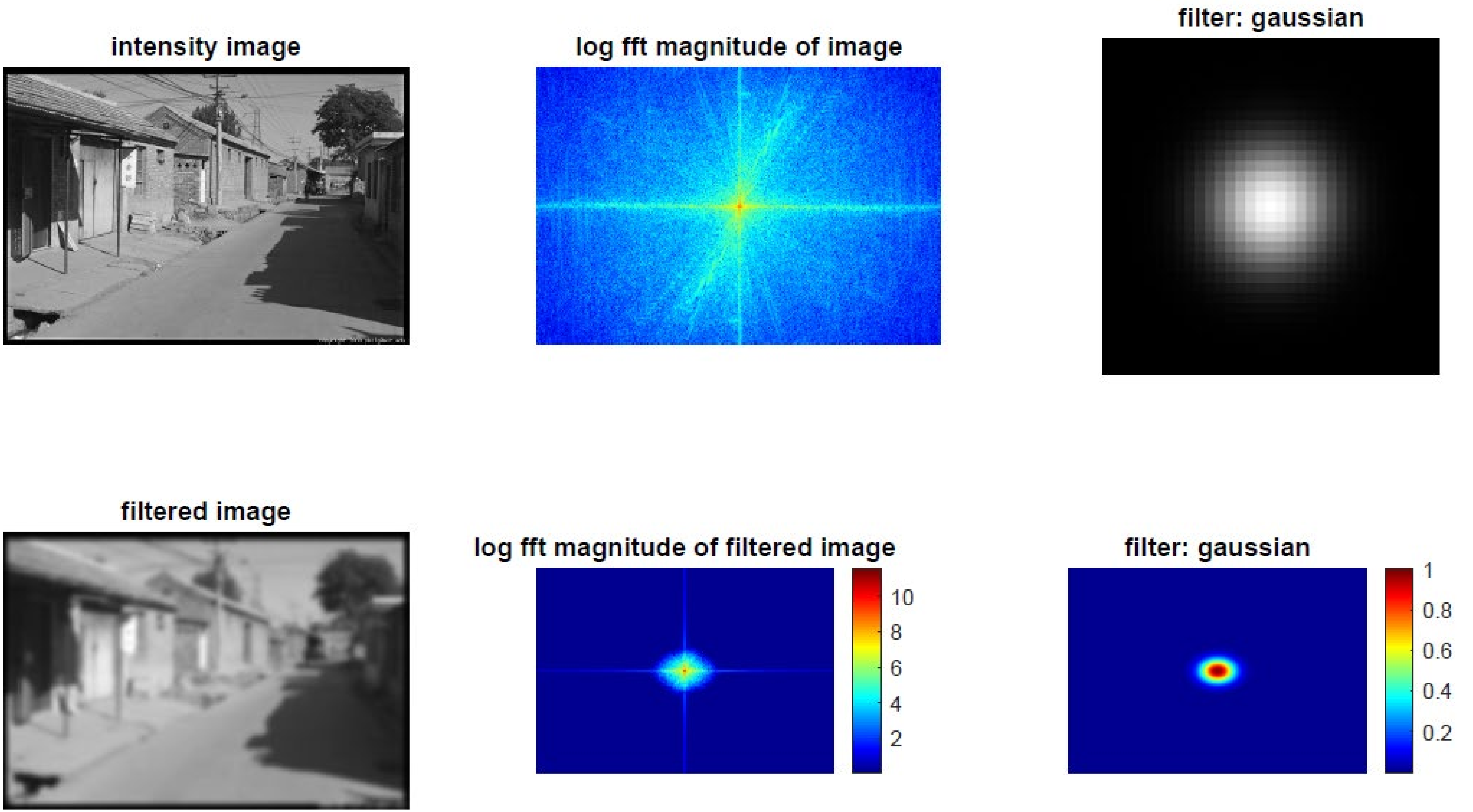
intensity image

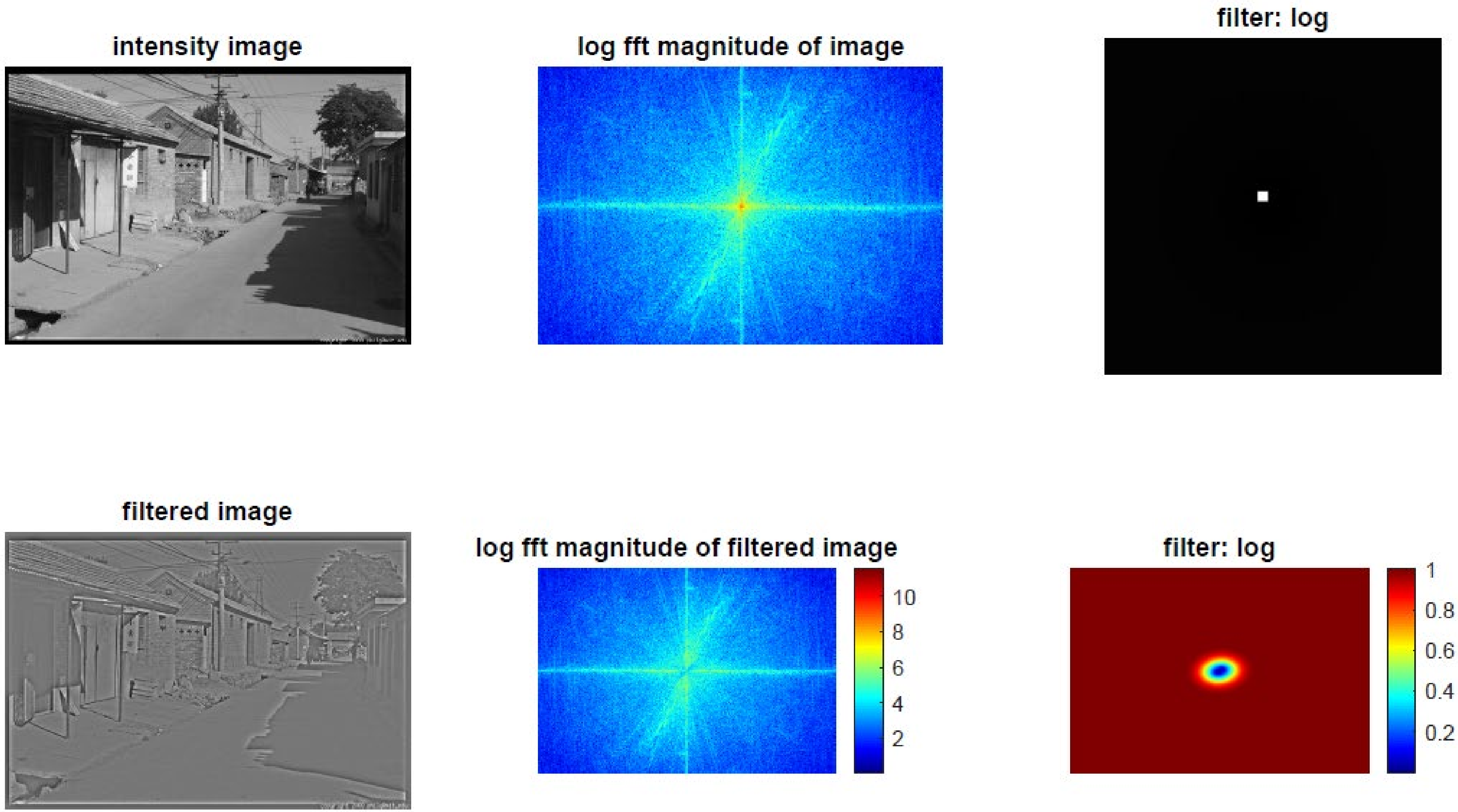
?

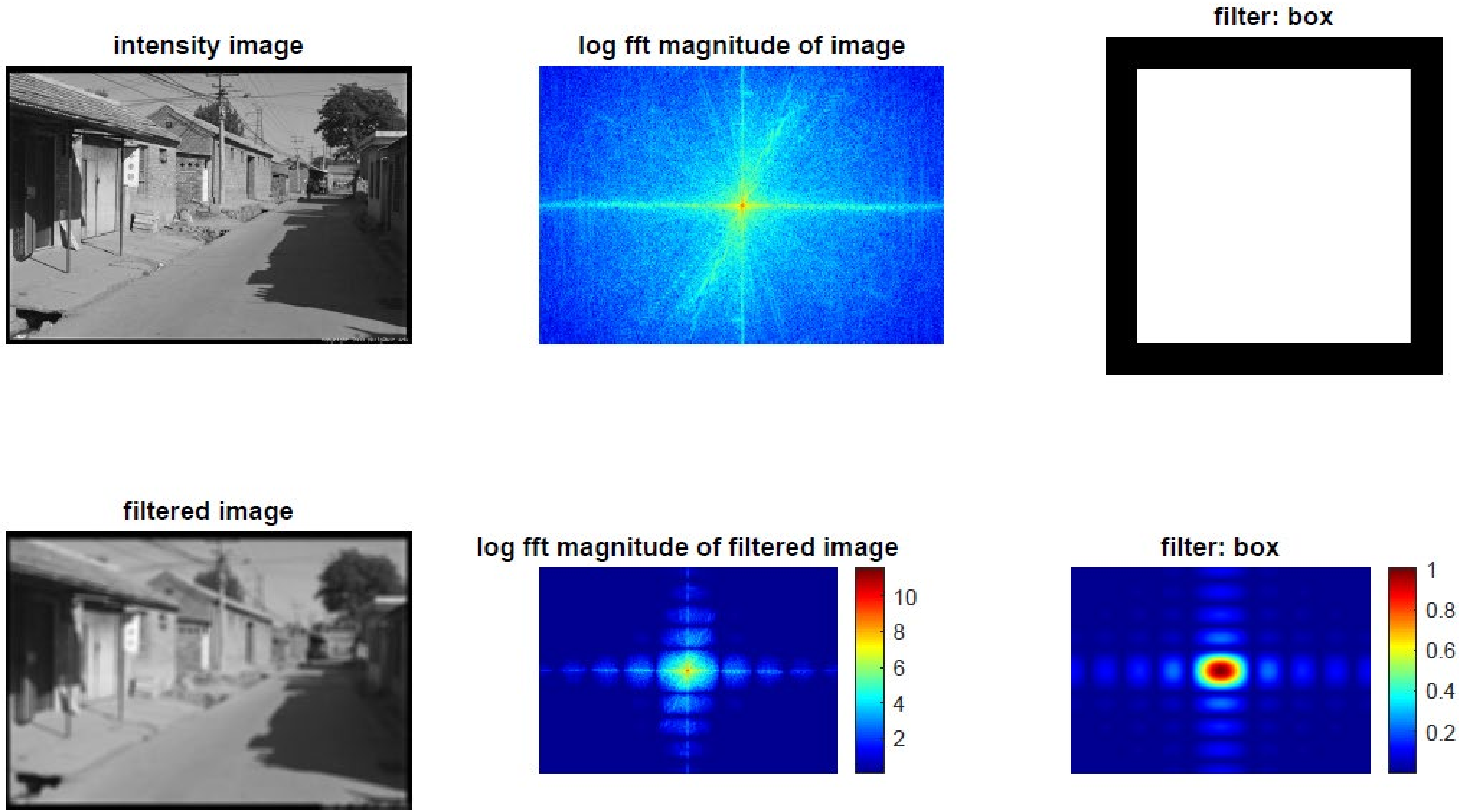
intensity image

?


intensity image
log fft magnitude of image
filter: sobel
filtered image
log fft magnitude of filtered image
10
8
6
4
2
filter: sobel
8
6
4
2


intensity image
log fft magnitude of image
filter: gaussian
filtered image
log fft magnitude of filtered image
10
8
6
4
2
filter: gaussian
1
0.8
0.6
0.4
0.2


intensity image
log fft magnitude of image
filter: log
filtered image
log fft magnitude of filtered image
10
8
6
4
2
filter: log
1
0.8
0.6
0.4
0.2


intensity image
log fft magnitude of image
filter: box
filtered image
log fft magnitude of filtered image
10
8
6
4
2
filter: box
1
0.8
0.6
0.4
0.2

FFT in Python

- Filtering with fft

```
import matplotlib.pyplot as plt
import numpy as np

def filter_image(im, fil):
    '''
    im: H x W floating point numpy ndarray representing image in grayscale
    fil: M x M floating point numpy ndarray representing 2D filter
    '''
    H, W = im.shape
    hs = fil.shape[0] // 2 # half of filter size
    fftsize = 1024         # should be order of 2 (for speed) and include padding
    im_fft = np.fft.fft2(im, (fftsize, fftsize))   # 1) fft im with padding
    fil_fft = np.fft.fft2(fil, (fftsize, fftsize)) # 2) fft fil, pad to same size as image
    im_fil_fft = im_fft * fil_fft;                 # 3) multiply fft images
    im_fil = np.fft.ifft2(im_fil_fft)              # 4) inverse fft2
    im_fil = im_fil[hs:hs + H, hs:hs + W]          # 5) remove padding
    im_fil = np.real(im_fil)                       # 6) extract out real part
    return im_fil
```

FFT in Python

- Displaying with fft

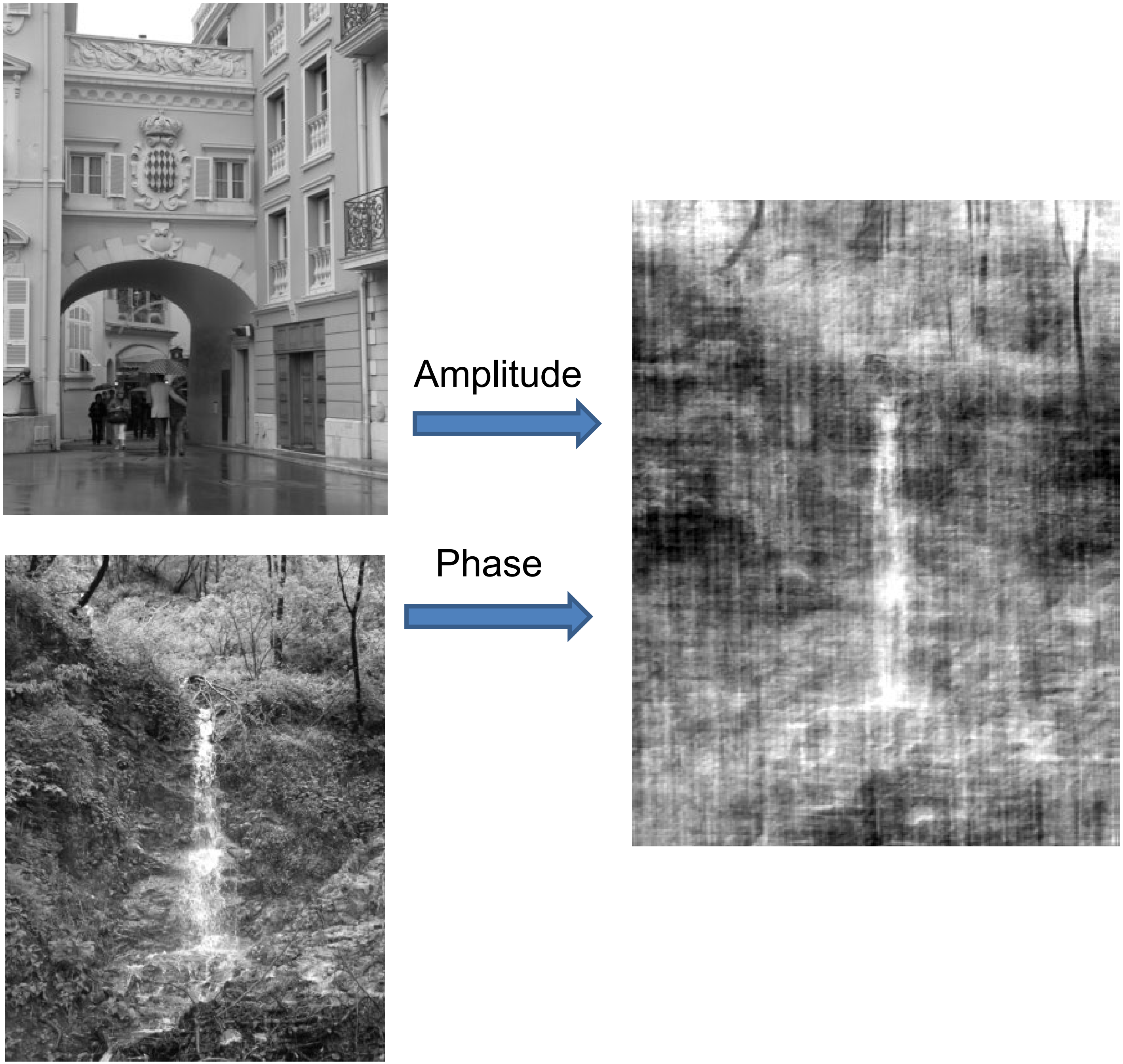
```
import matplotlib.pyplot as plt
import numpy as np

def display_frequency_image(frequency_image):
    '''
    frequency_image: H x W floating point numpy ndarray representing image after FFT
                     in grayscale
    '''
    shifted_image = np.fft.fftshift(frequency_image)
    amplitude_image = np.abs(shifted_image)
    log_amplitude_image = np.log(amplitude_image)
    fig = plt.figure()
    plt.imshow(log_amplitude_image, cmap='gray')
    plt.show()
```

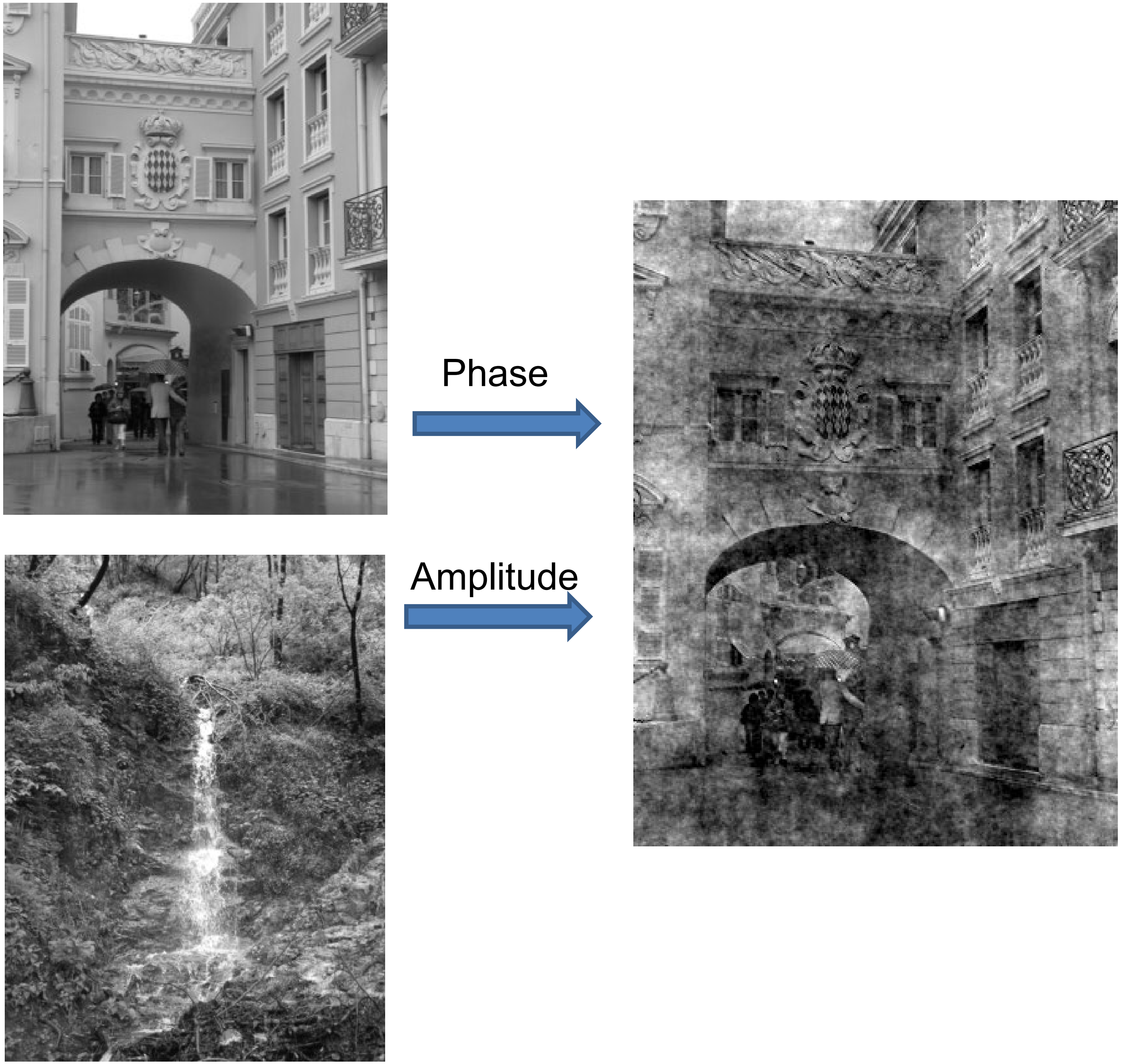
Questions

Which has more information, the phase or the magnitude?

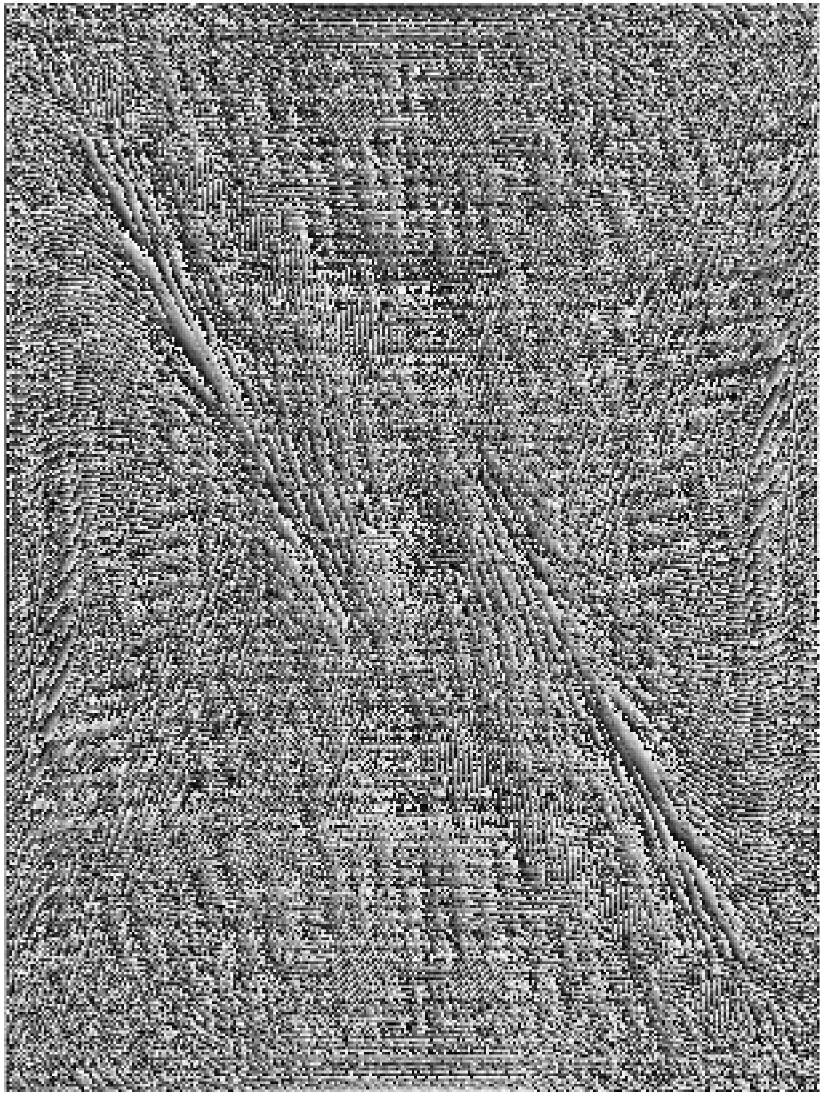
What happens if you take the phase from one image and combine it with the magnitude from another image?

```
%% Compute FFT and decompose to magnitude and phase
im1_fft = fft2(im1);
im1_fft_mag = abs(im1_fft);
im1_fft_phase = angle(im1_fft);

im2_fft = fft2(im2);
im2_fft_mag = abs(im2_fft);
im2_fft_phase = angle(im2_fft);

%% Combine mag and phase from different images and compute inverse FFT
mag1_phase2 = ifft2(im1_fft_mag.*cos(im2_fft_phase)+1i*im1_fft_mag.*sin(im2_fft_phase));
phase1_mag2 =ifft2(im2_fft_mag.*cos(im1_fft_phase)+1i*im2_fft_mag.*sin(im1_fft_phase));
```


Amplitude
Phase

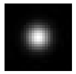

Phase
Amplitude

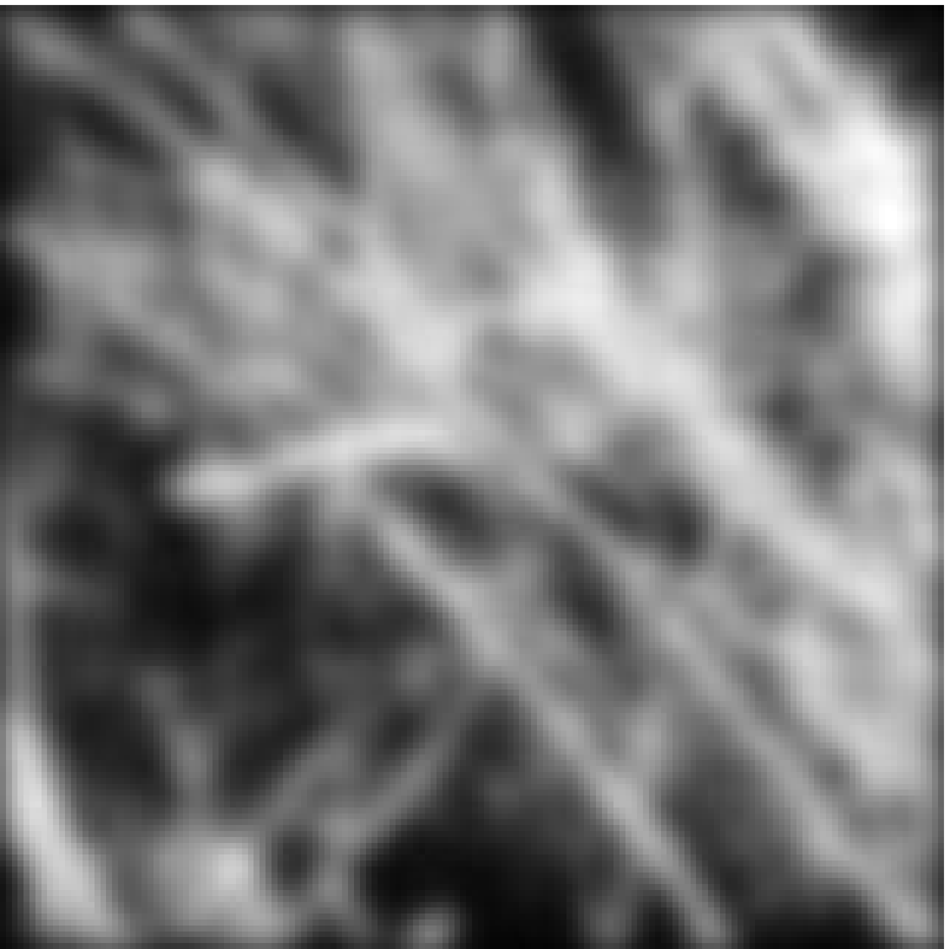
Phase Image

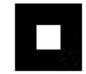


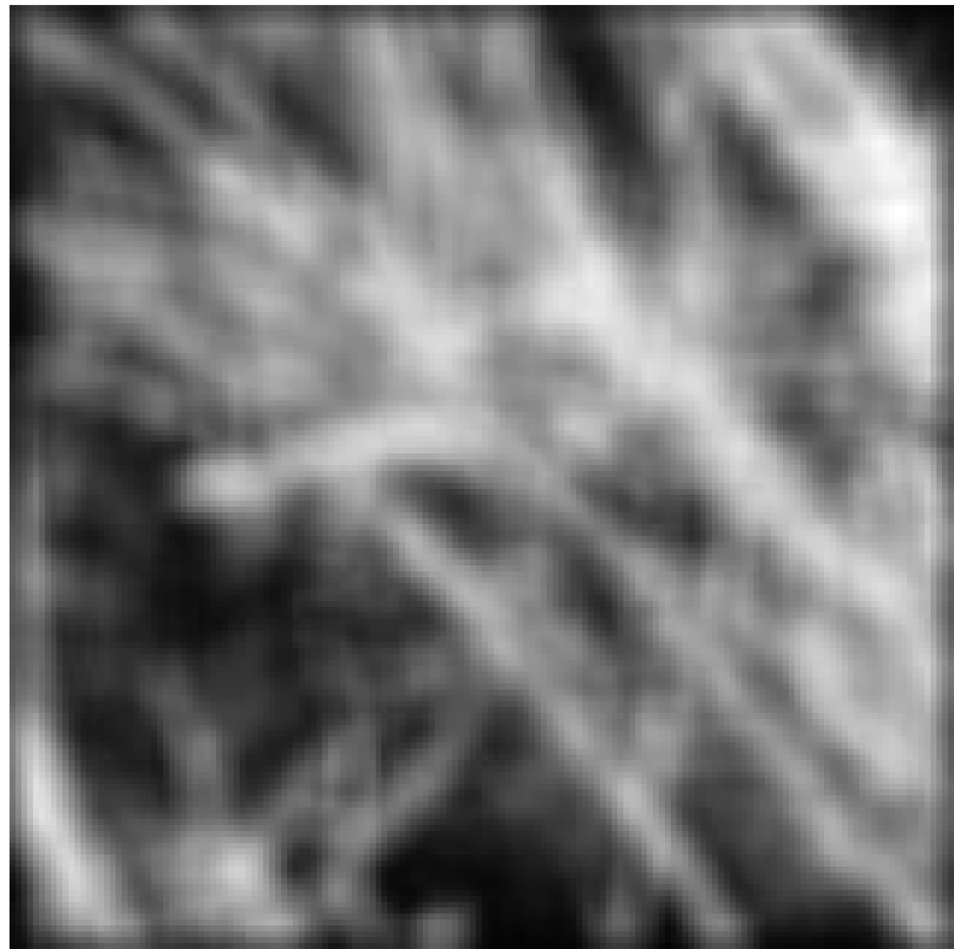
Filtering

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

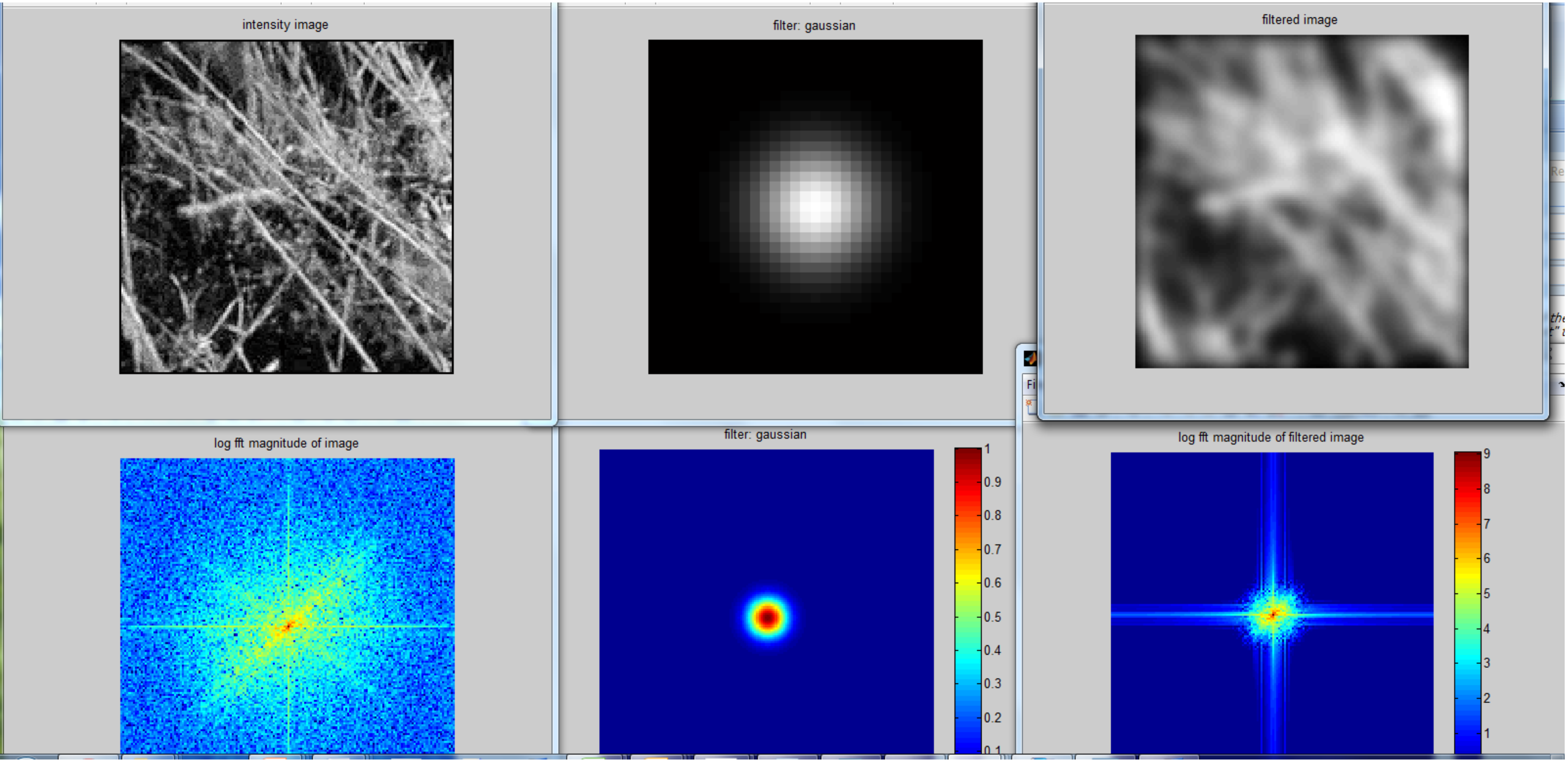
Gaussian 



Box filter 

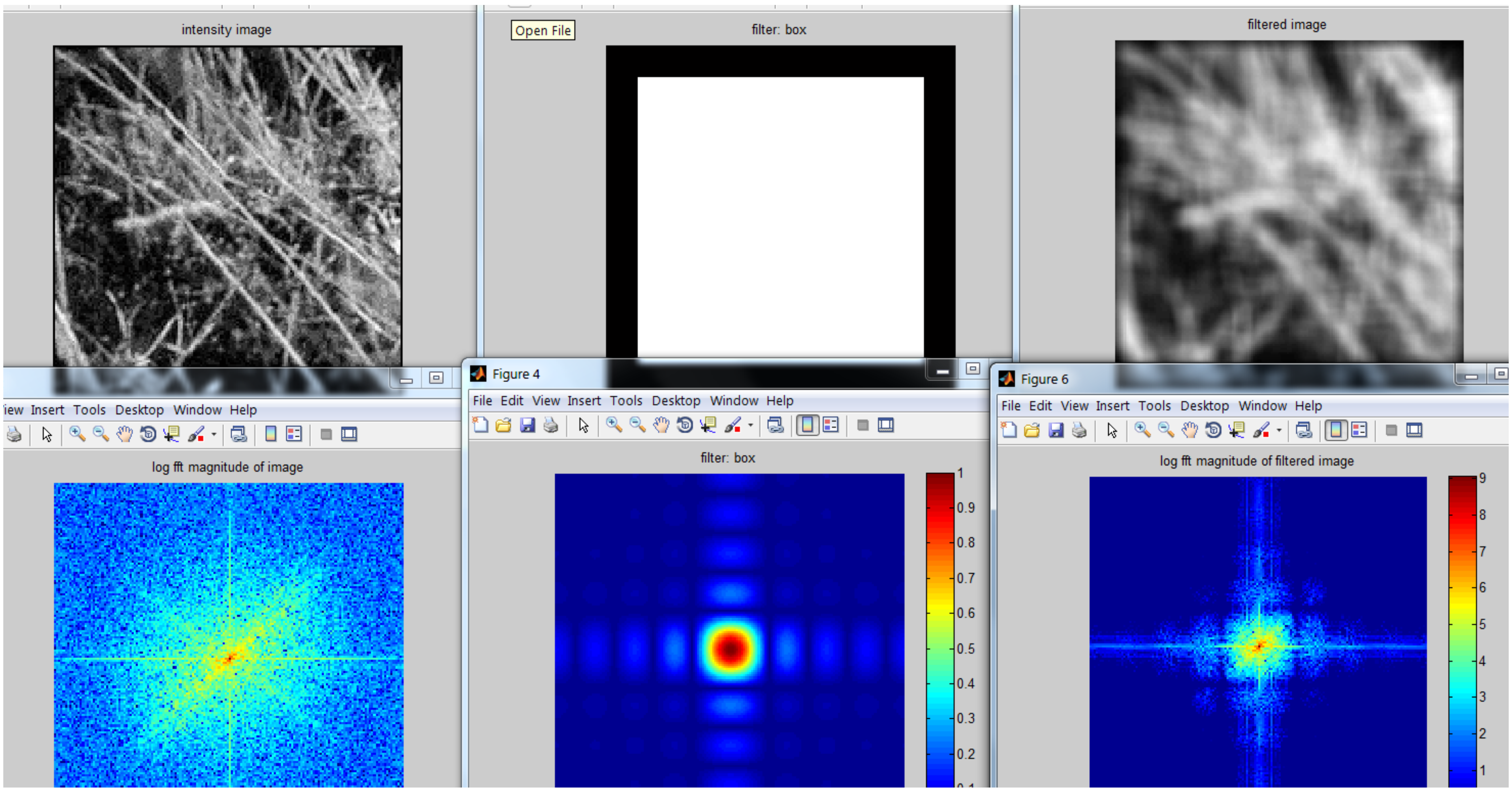


Gaussian



intensity image
filter: gaussian
filtered image
log fft magnitude of image
filter: gaussian
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
log fft magnitude of filtered image
9
8
7
6
5
4
3
2
1

Box Filter



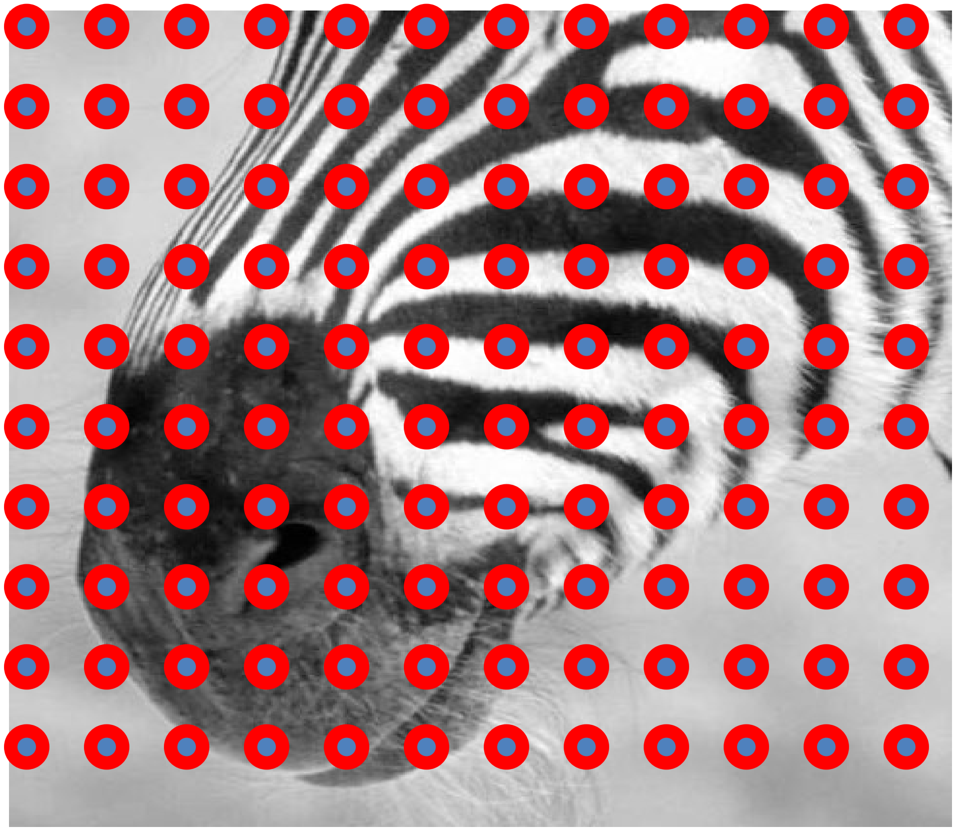
intensity image
Open File
filter: box
filtered image
Figure 4
Figure 6
File Edit View Insert Tools Desktop Window Help
log fft magnitude of image
filter: box
log fft magnitude of filtered image

Sampling

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

Image: http://www.flickr.com/photos/igorms/136916757/

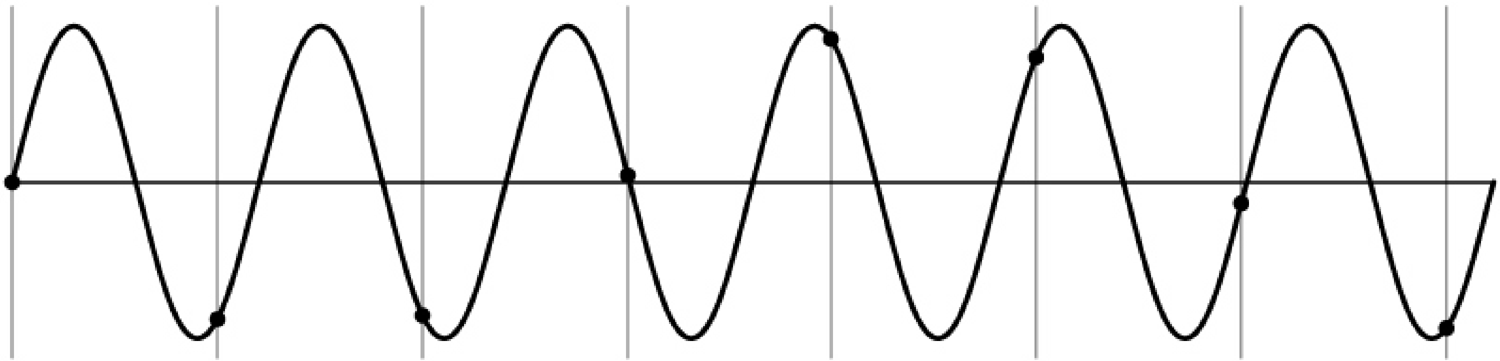
Subsampling by a factor of 2



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

Aliasing problem

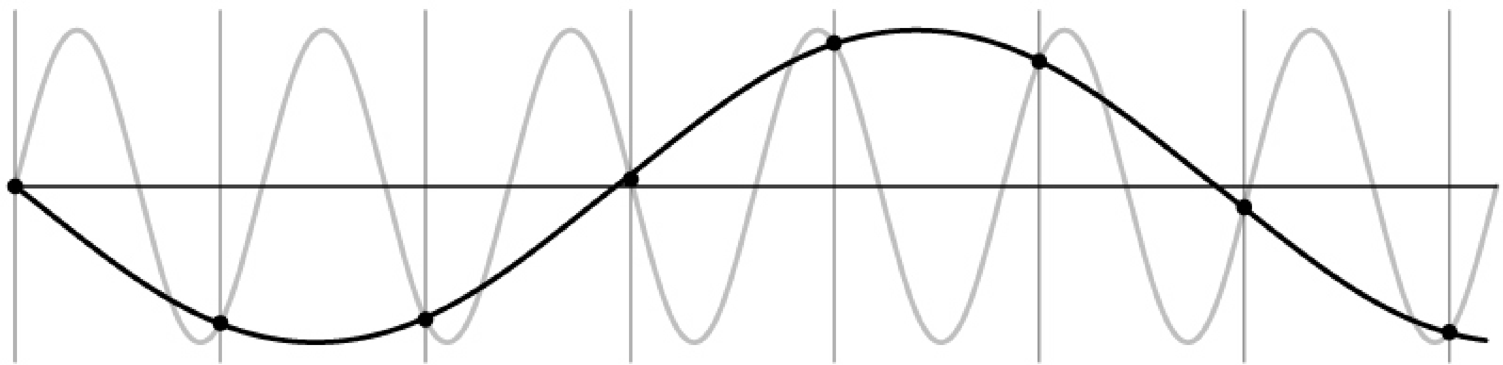
- 1D example (sinewave):



Source: S. Marschner

Aliasing problem

- 1D example (sinewave):



Source: S. Marschner

Aliasing problem

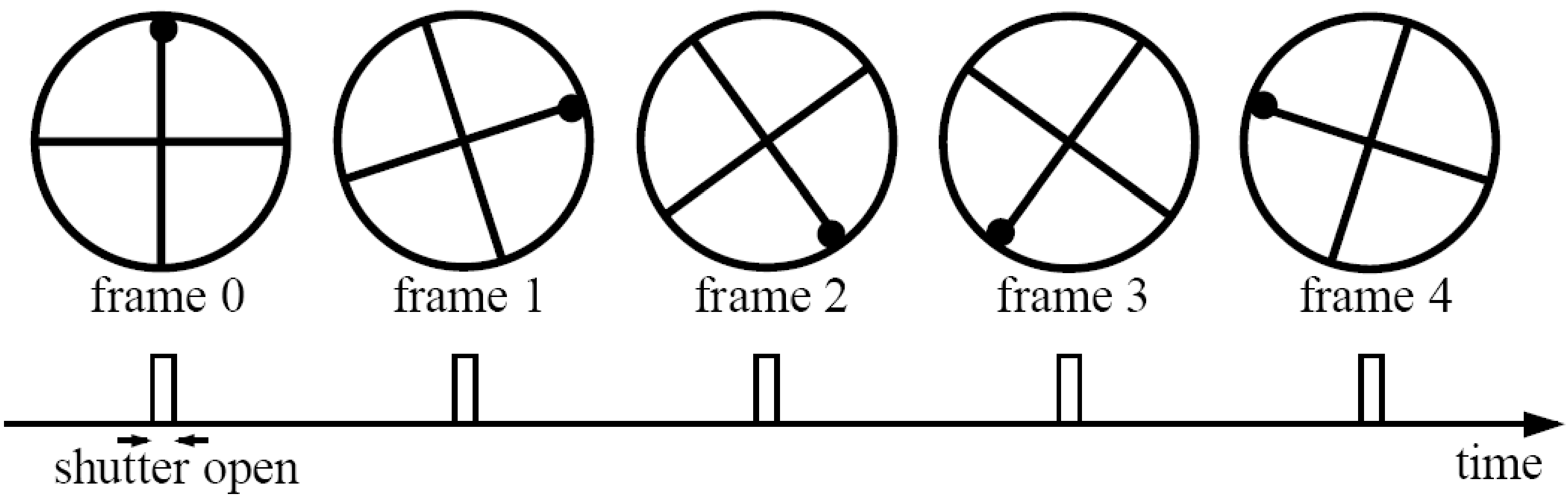
- Sub-sampling may be dangerous….
- Characteristic errors may appear:
  - “Wagon wheels rolling the wrong way in movies”
  - “Checkerboards disintegrate in ray tracing”
  - “Striped shirts look funny on color television”

Source: D. Forsyth

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):



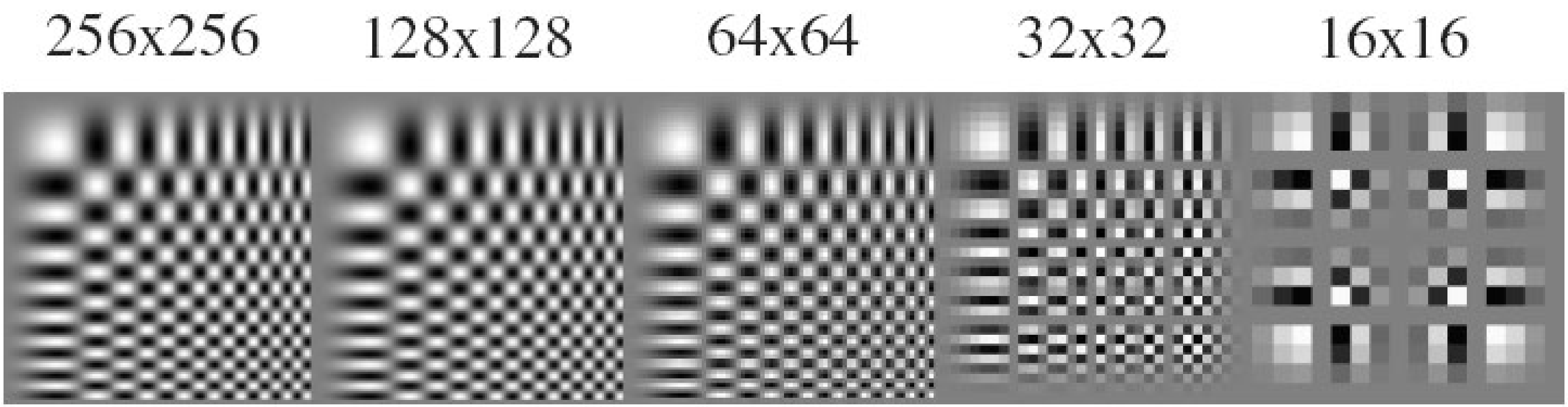
Without dot, wheel appears to be rotating slowly backwards! (counterclockwise)

Slide by Steve Seitz

Aliasing in graphics

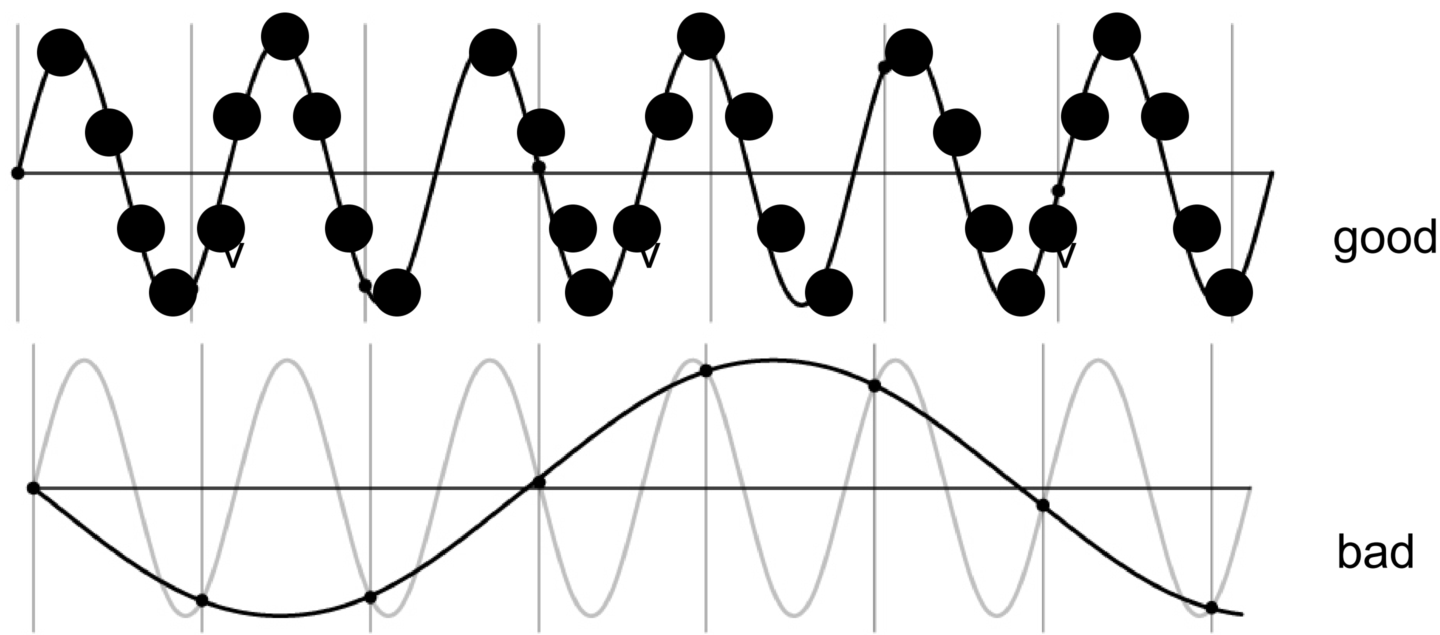
Source: A. Efros

Sampling and aliasing



Nyquist-Shannon Sampling Theorem

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



Anti-aliasing

Solutions:

- Sample more often

- 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 filter

Algorithm for downsampling by factor of 2

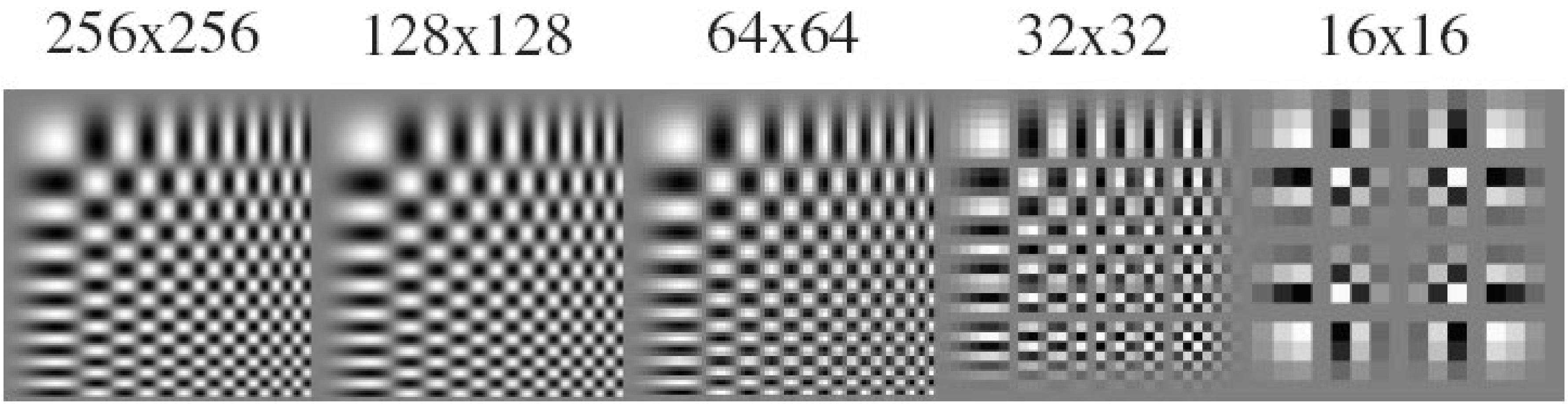
1. Start with image(h, w)
2. Apply low-pass filter

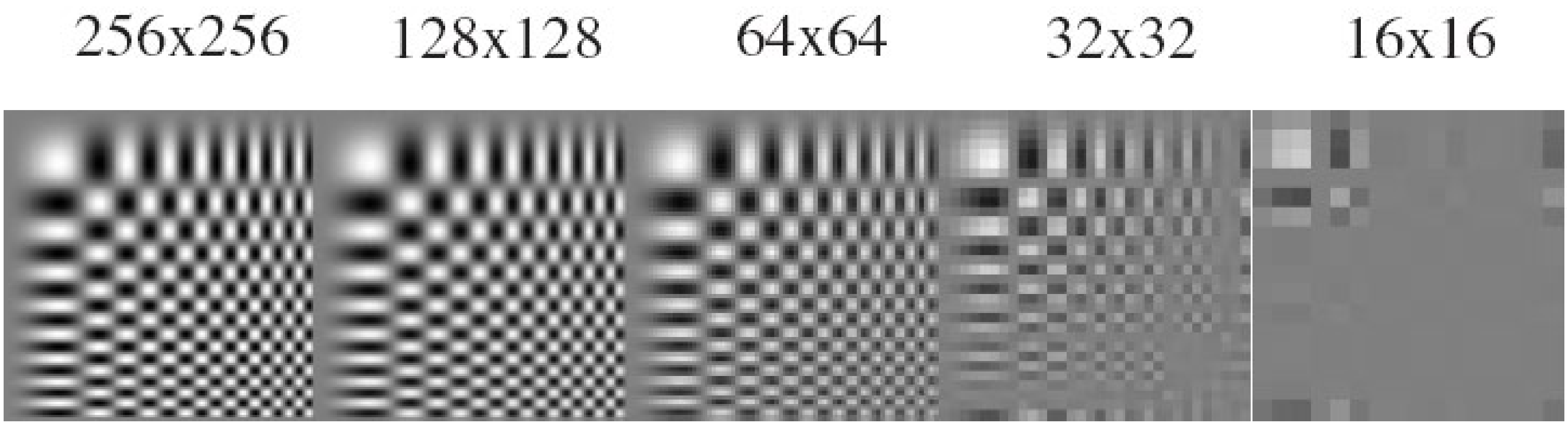
    ```
    im_blur = imfilter(image, fspecial('gaussian', 7, 1))
    ```

3. Sample every other pixel

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

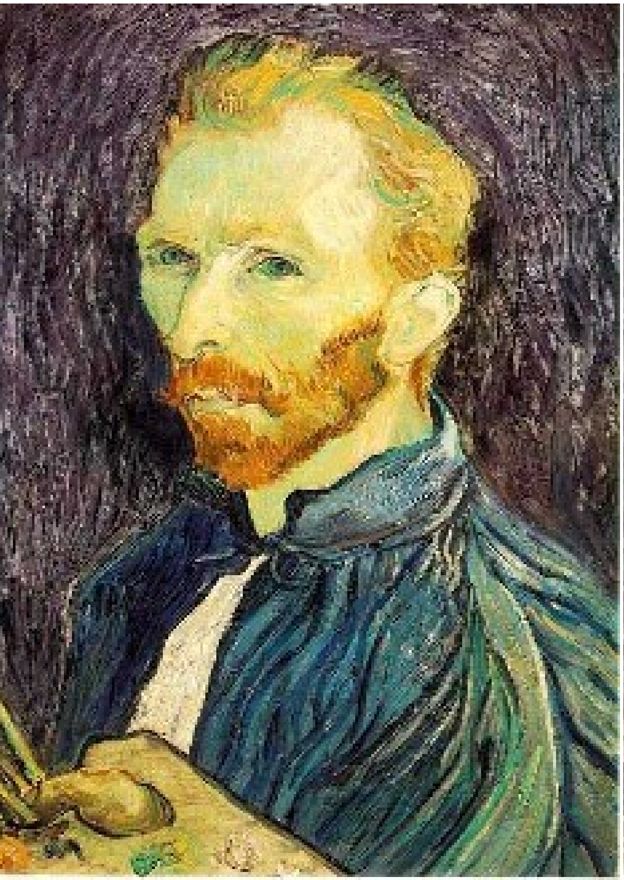
Anti-aliasing



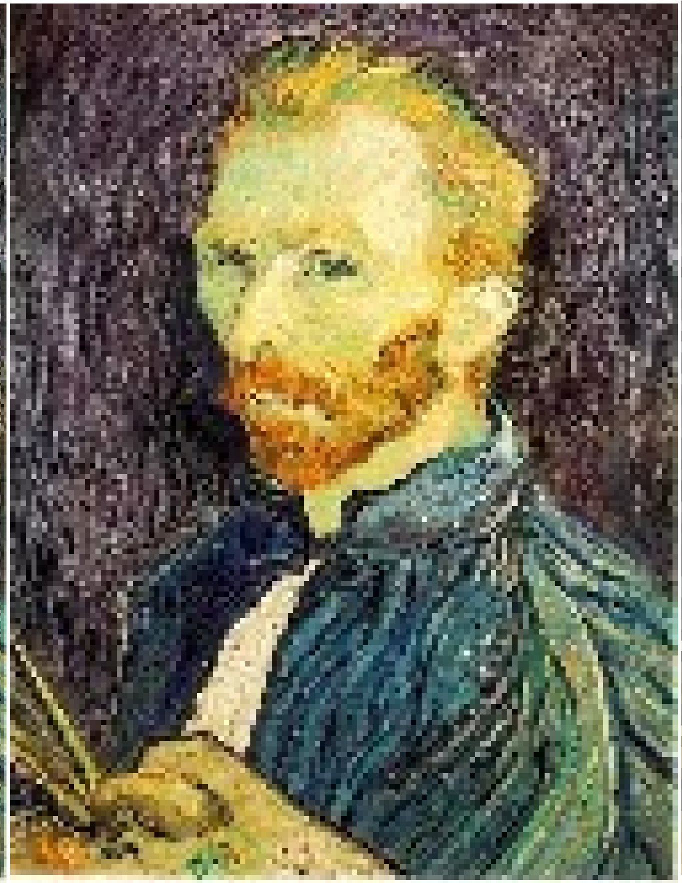


Forsyth and Ponce 2002

Subsampling without pre-filtering



1/2

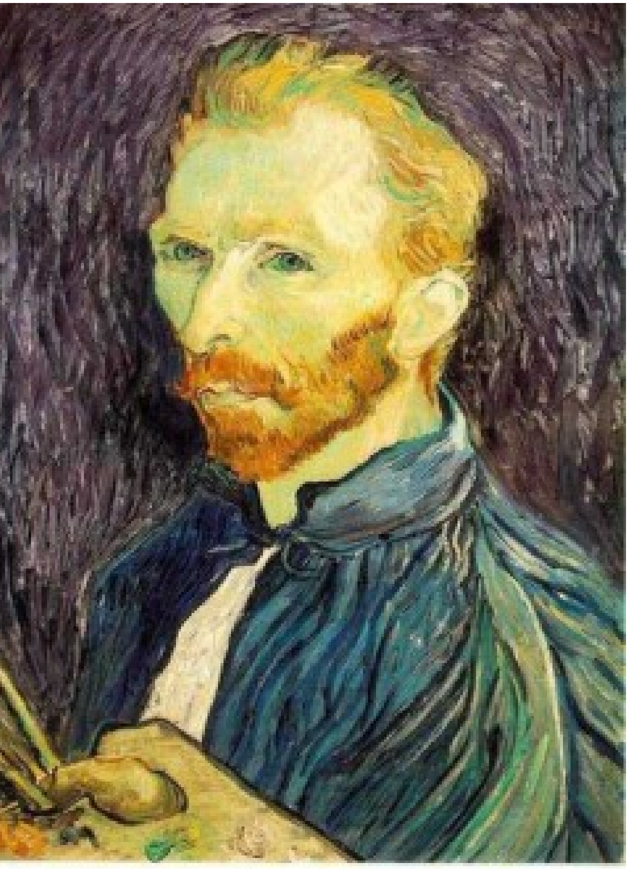


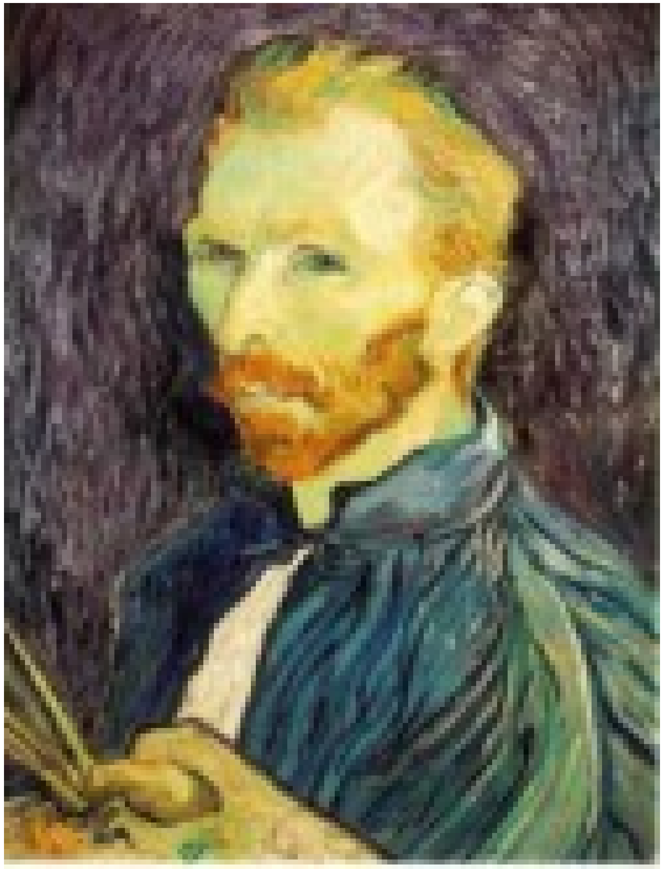
1/4 (2x zoom)

1/8 (4x zoom)

Slide by Steve Seitz

Subsampling with Gaussian pre-filtering


Gaussian 1/2


G 1/4

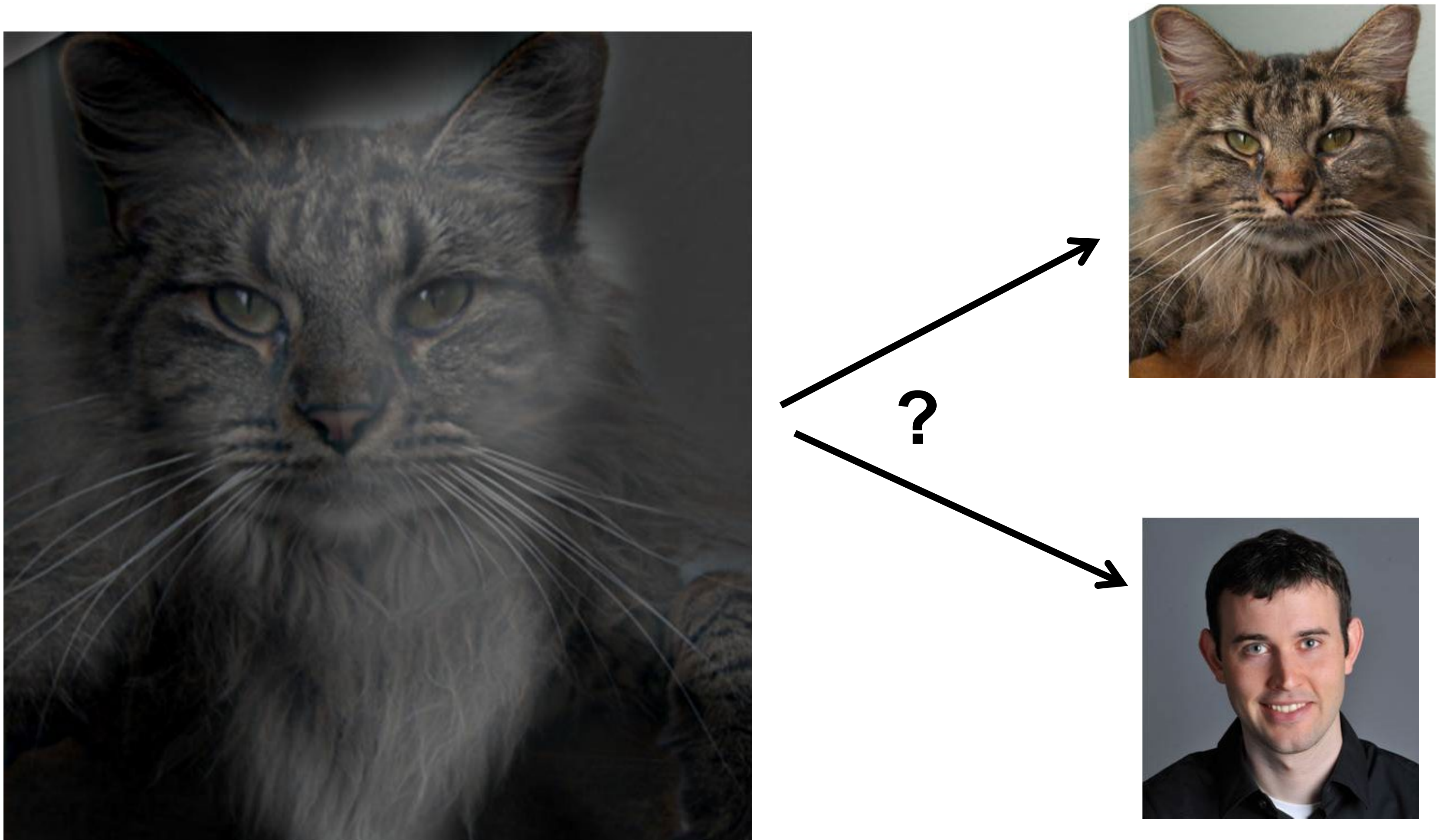

G 1/8

Slide by Steve Seitz

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

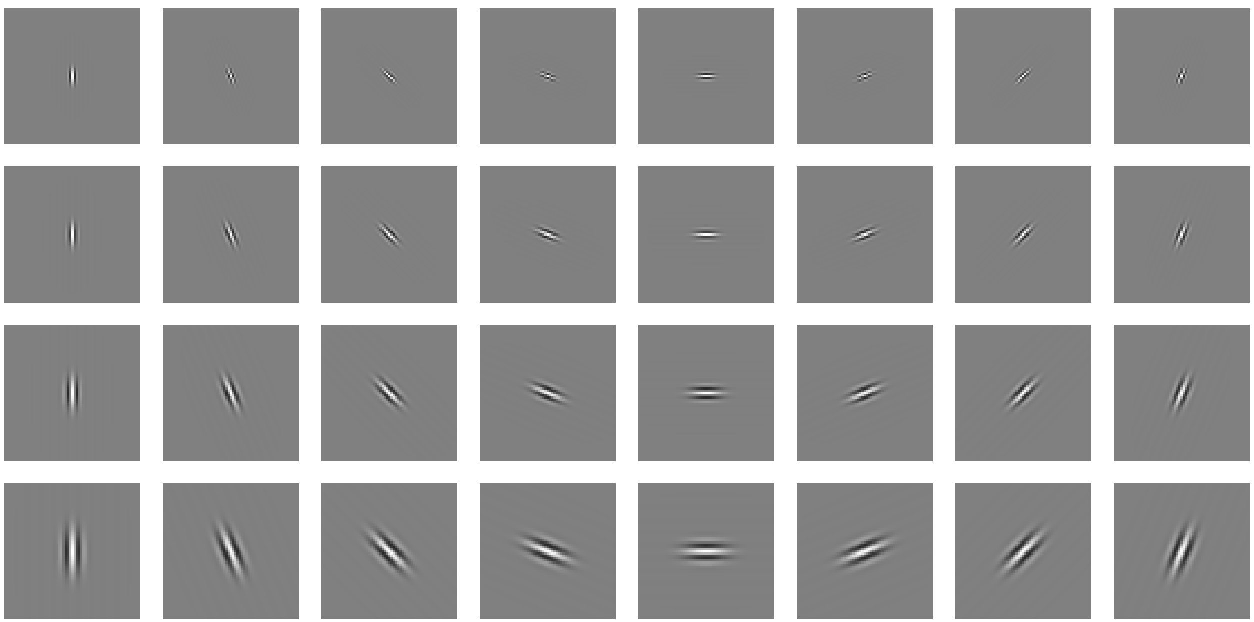
Image: http://www.flickr.com/photos/igorms/136916757/

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



Clues from Human Perception

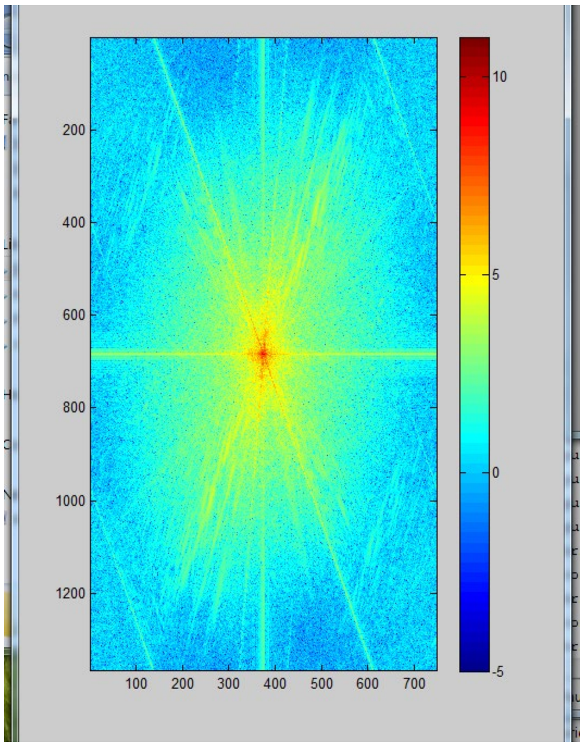
- Early processing in humans filters for various orientations and scales of frequency
- Perceptual cues in the mid frequencies dominate perception
- When we see an image from far away, we are effectively subsampling it



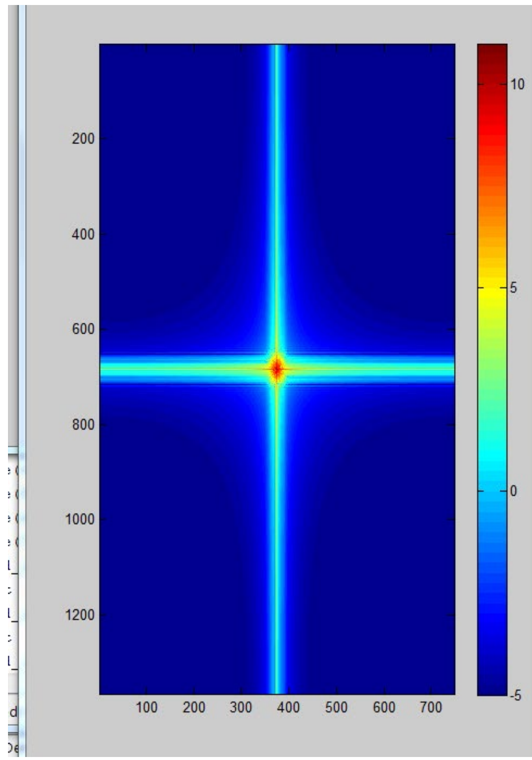
Early Visual Processing: Multi-scale edge and blob filters

Hybrid Image in FFT

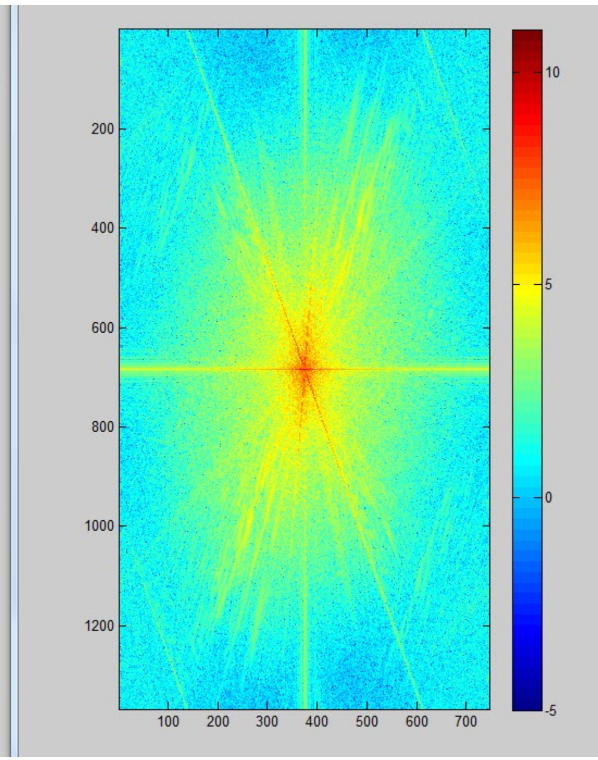
Hybrid Image



Low-passed Image

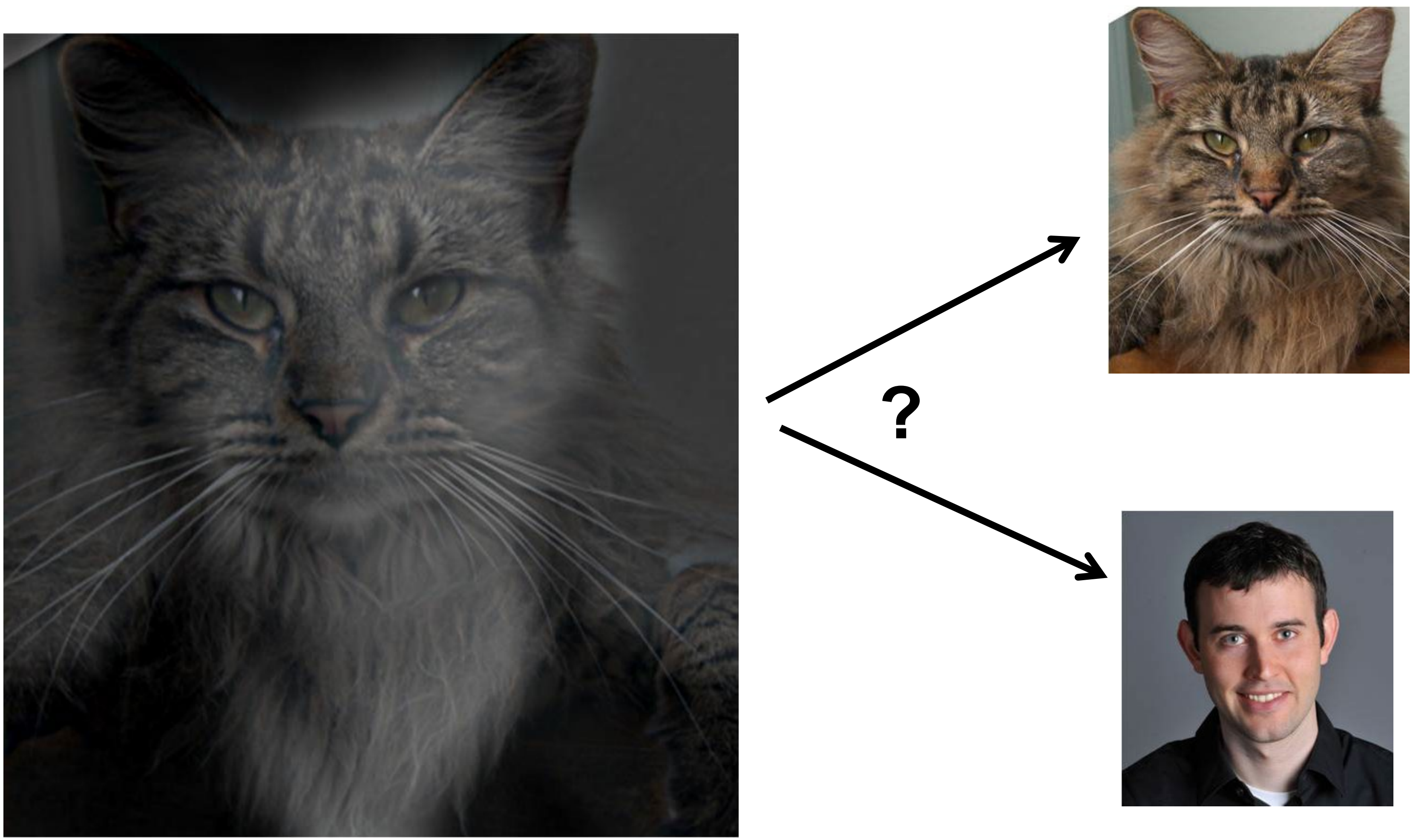


High-passed Image

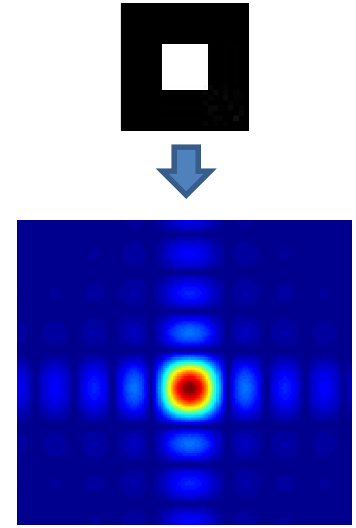


Perception

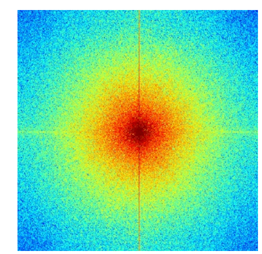
Why do we get different, distance-dependent interpretations of hybrid images?



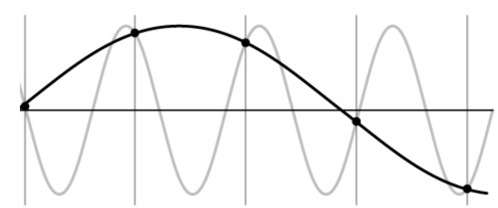
Things to Remember



- 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)



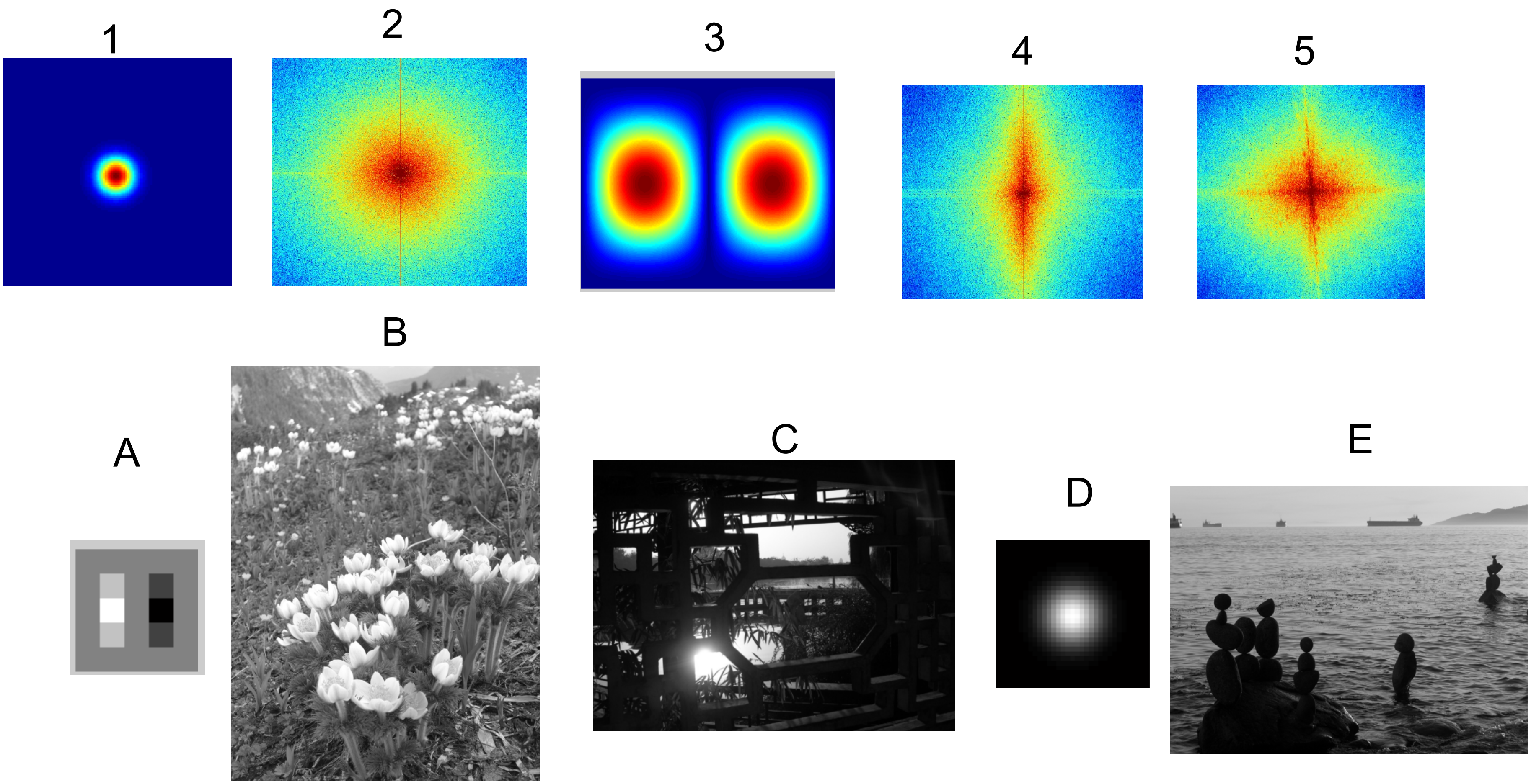
- Images are mostly smooth
  - Basis for compression



- Remember to low-pass before sampling

Take-home question

1. Match the spatial domain image to the Fourier magnitude image



Next class: applications of filtering

- Denoising
- Template matching
- Image pyramids
- Compression