



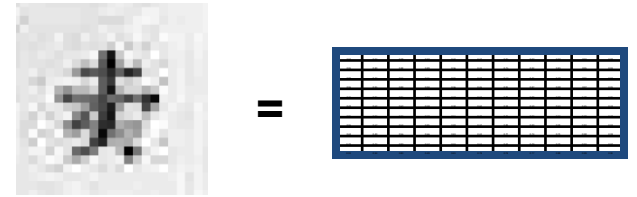
Thinking in Frequency

Computational Photography
University of Illinois

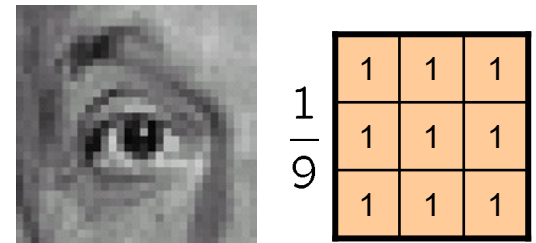
Derek Hoiem

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)



Review: questions

1. Write down a 3x3 filter that returns a positive value if the average value of the 4-adjacent neighbors is less than the center, and a zero or negative value otherwise
2. Write down a filter that will compute the gradient in the x-direction:

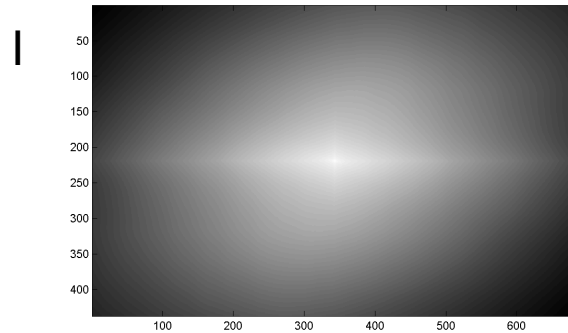
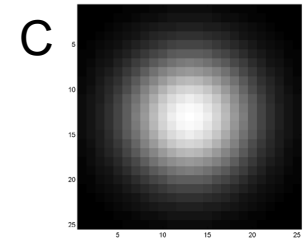
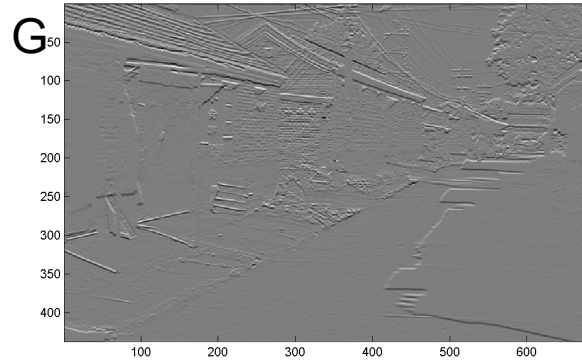
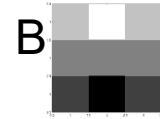
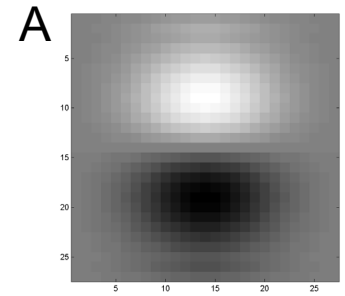
$$\text{gradx}[y, x] = \text{im}[y, x+1] - \text{im}[y, x] \text{ for each } x, y$$

Review: questions

3. Fill in the blanks:

- a) $_ = D * B$
 b) $A = _ * _$
 c) $F = D * _$
 d) $_ = D * D$

Filtering Operator \swarrow



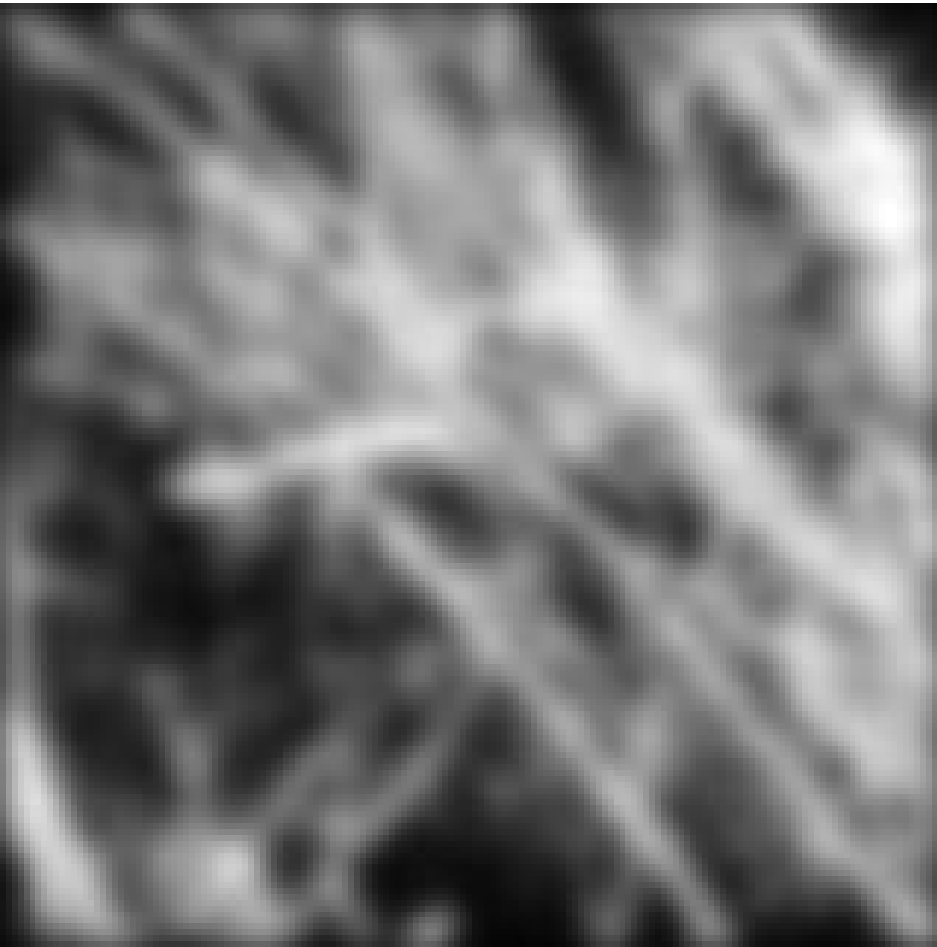
Today's class

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

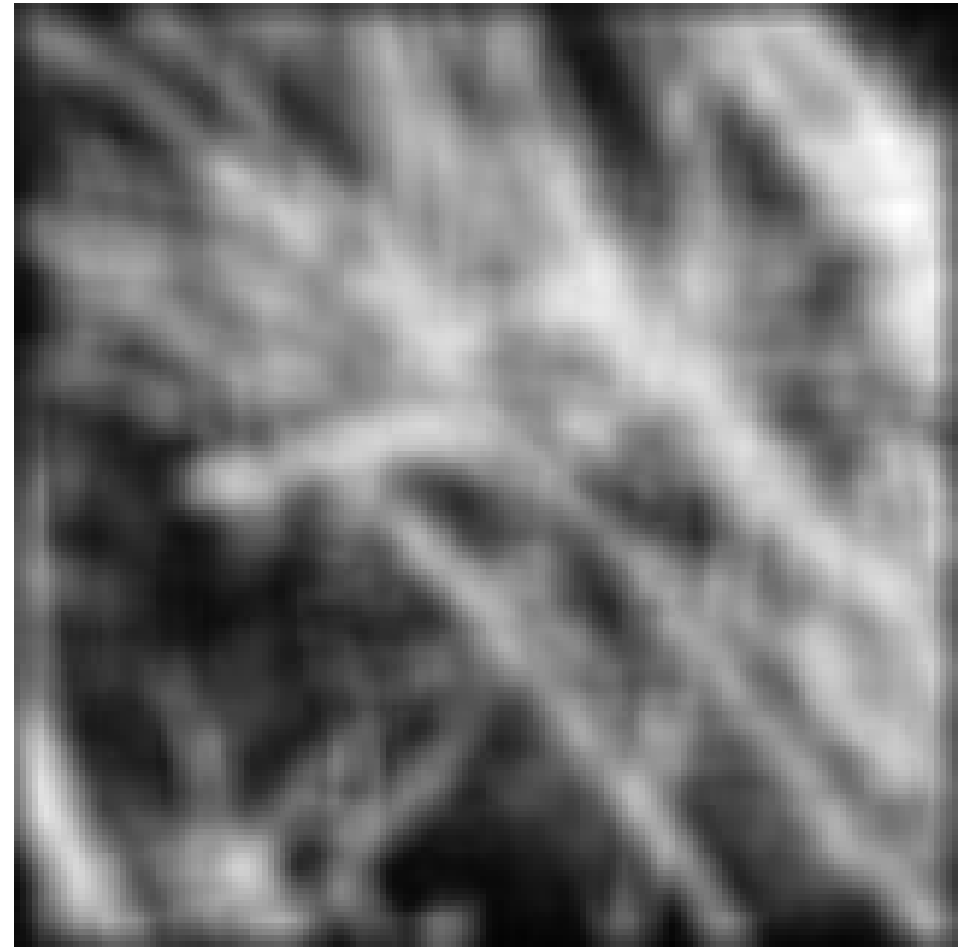
Why does the Gaussian give a nice smooth image, but the square filter give edgy artifacts?



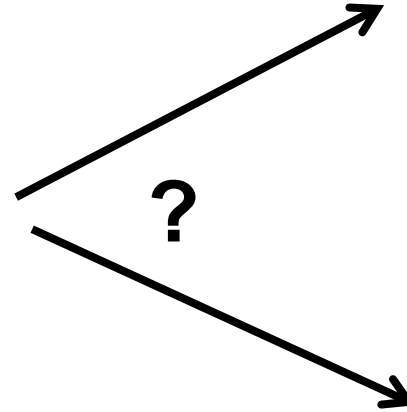
Gaussian



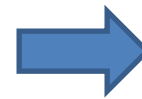
Box filter



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?



Thinking in terms of frequency

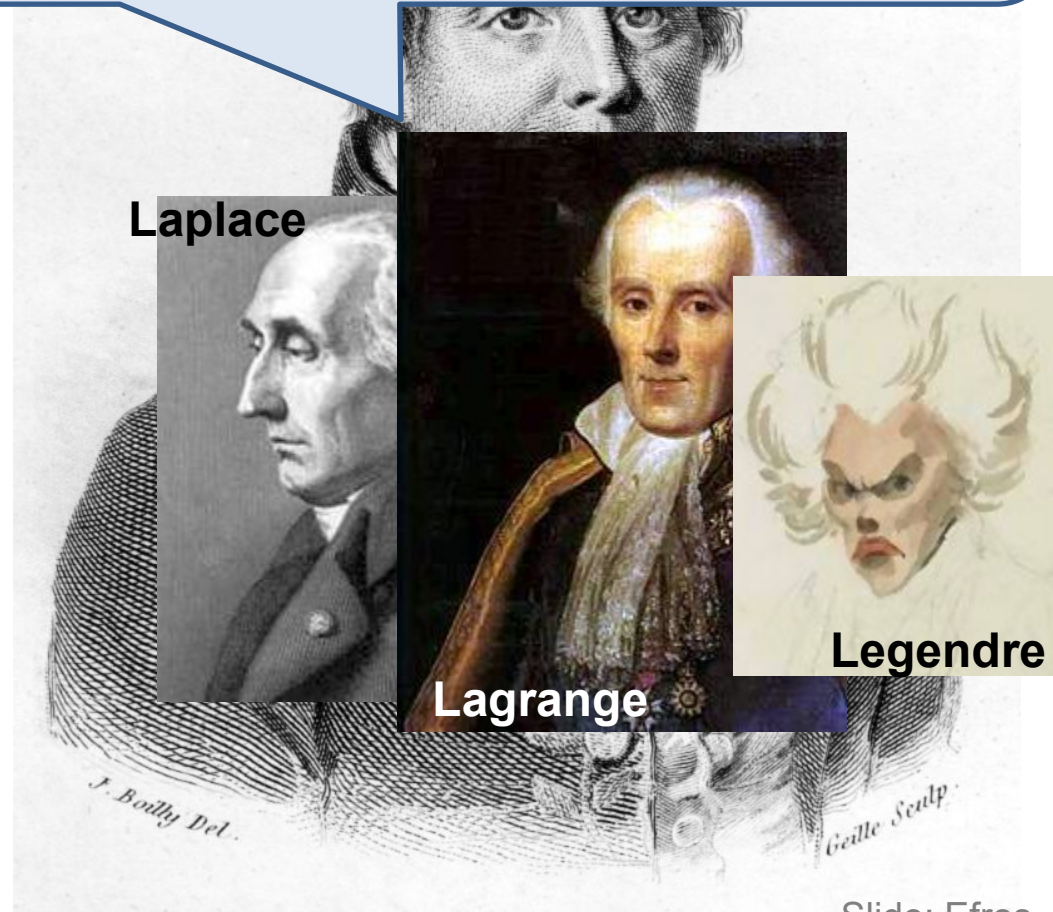
Jean Baptiste Joseph Fourier (1768-1830)

had crazy idea (1807):

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

...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.

- 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

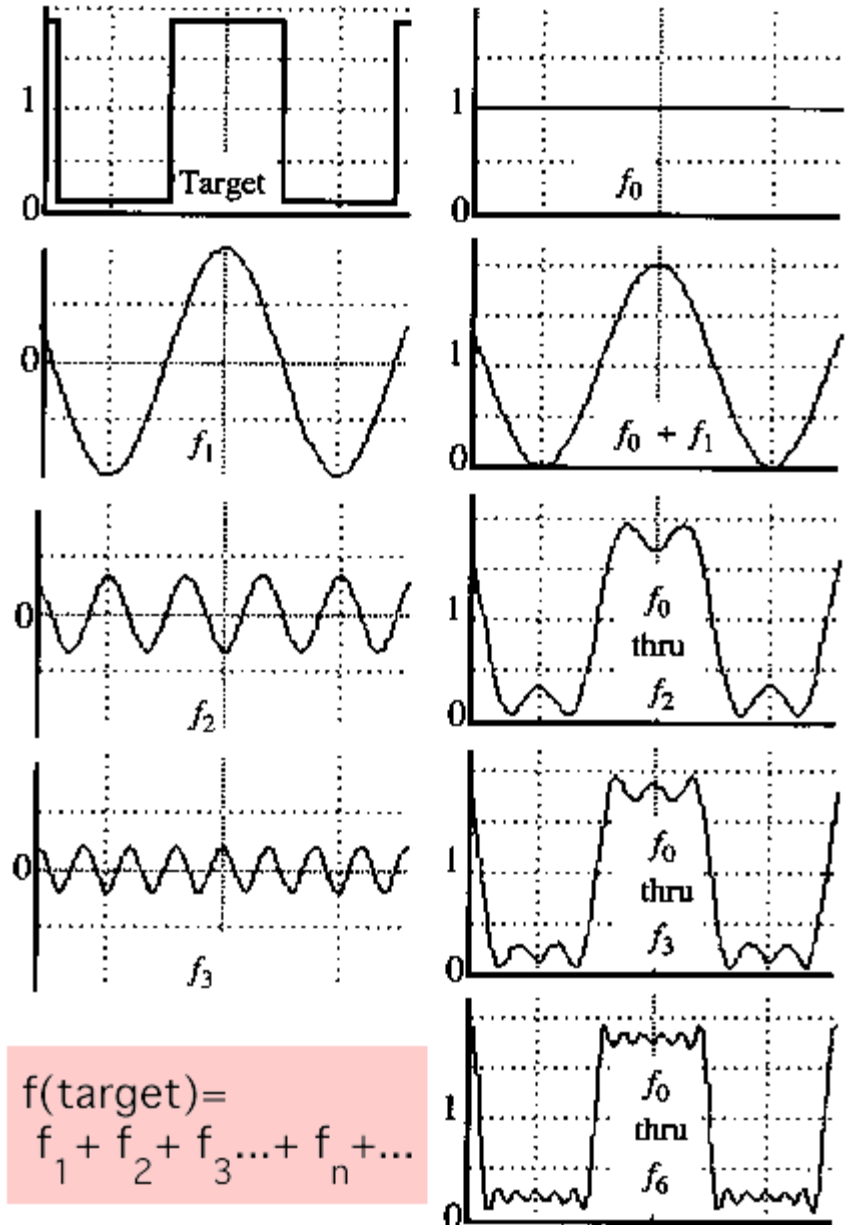


A sum of sines

Our building block:

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

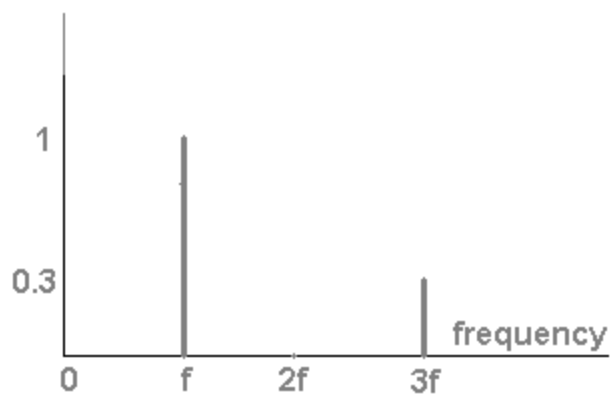
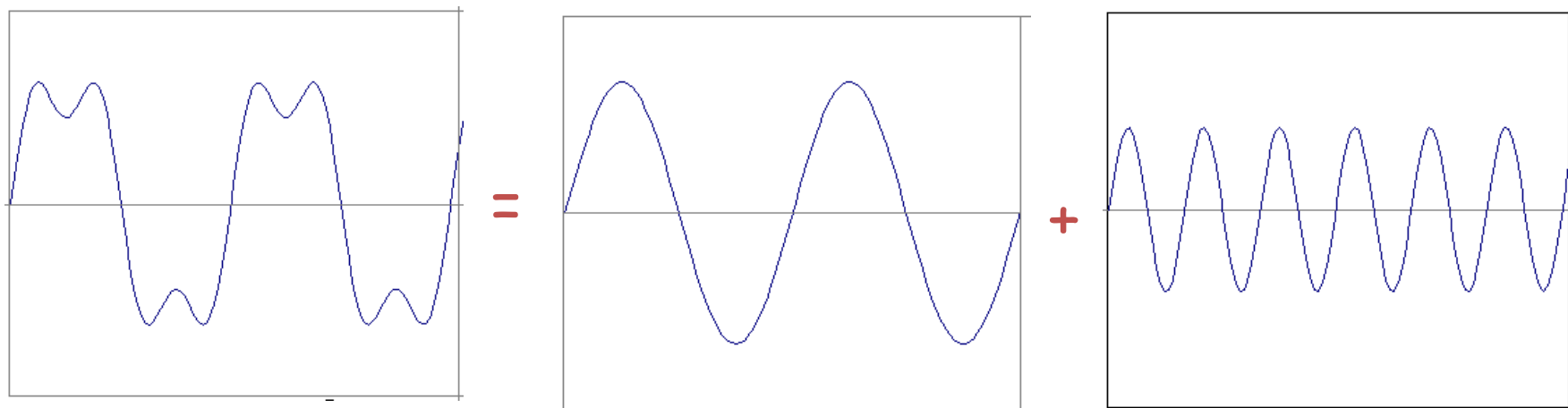
Add enough of them to get any signal $f(x)$ you want!



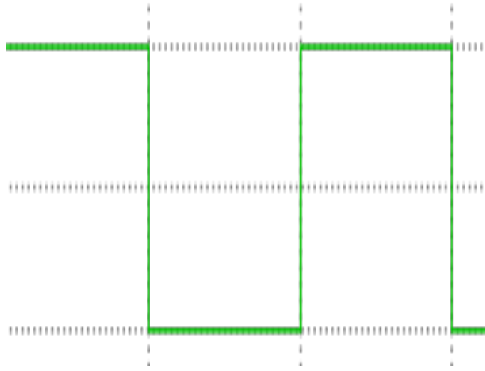
$$f(\text{target}) = f_1 + f_2 + f_3 + \dots + f_n + \dots$$

Frequency Spectra

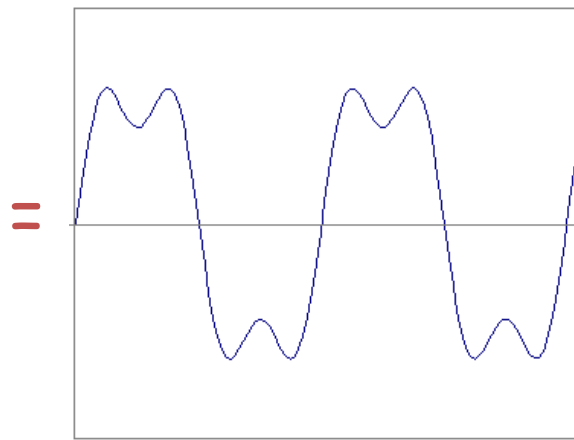
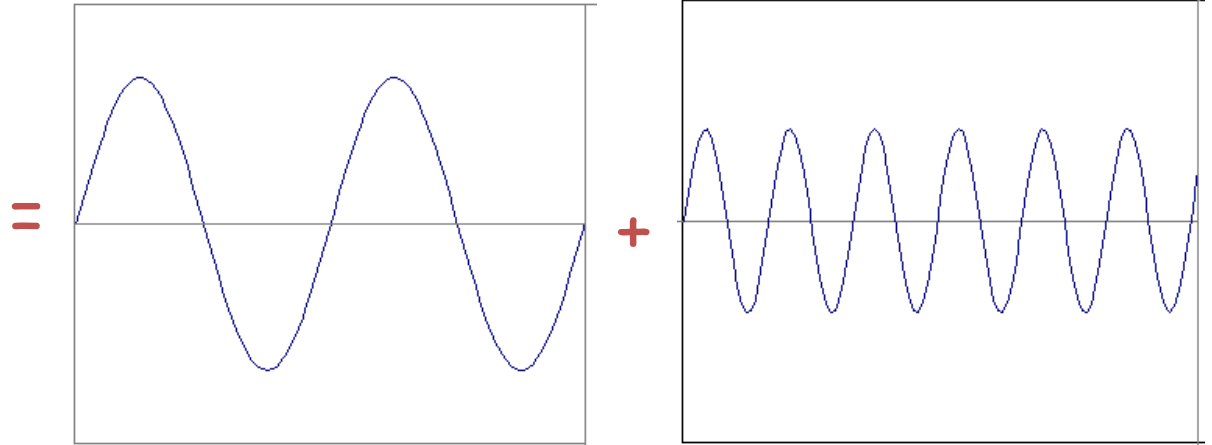
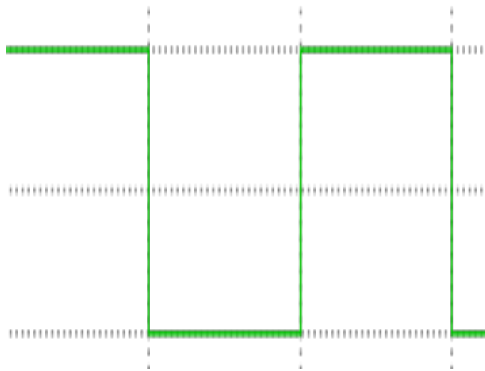
- example : $g(t) = \sin(2\pi f t) + (1/3)\sin(2\pi(3f) t)$



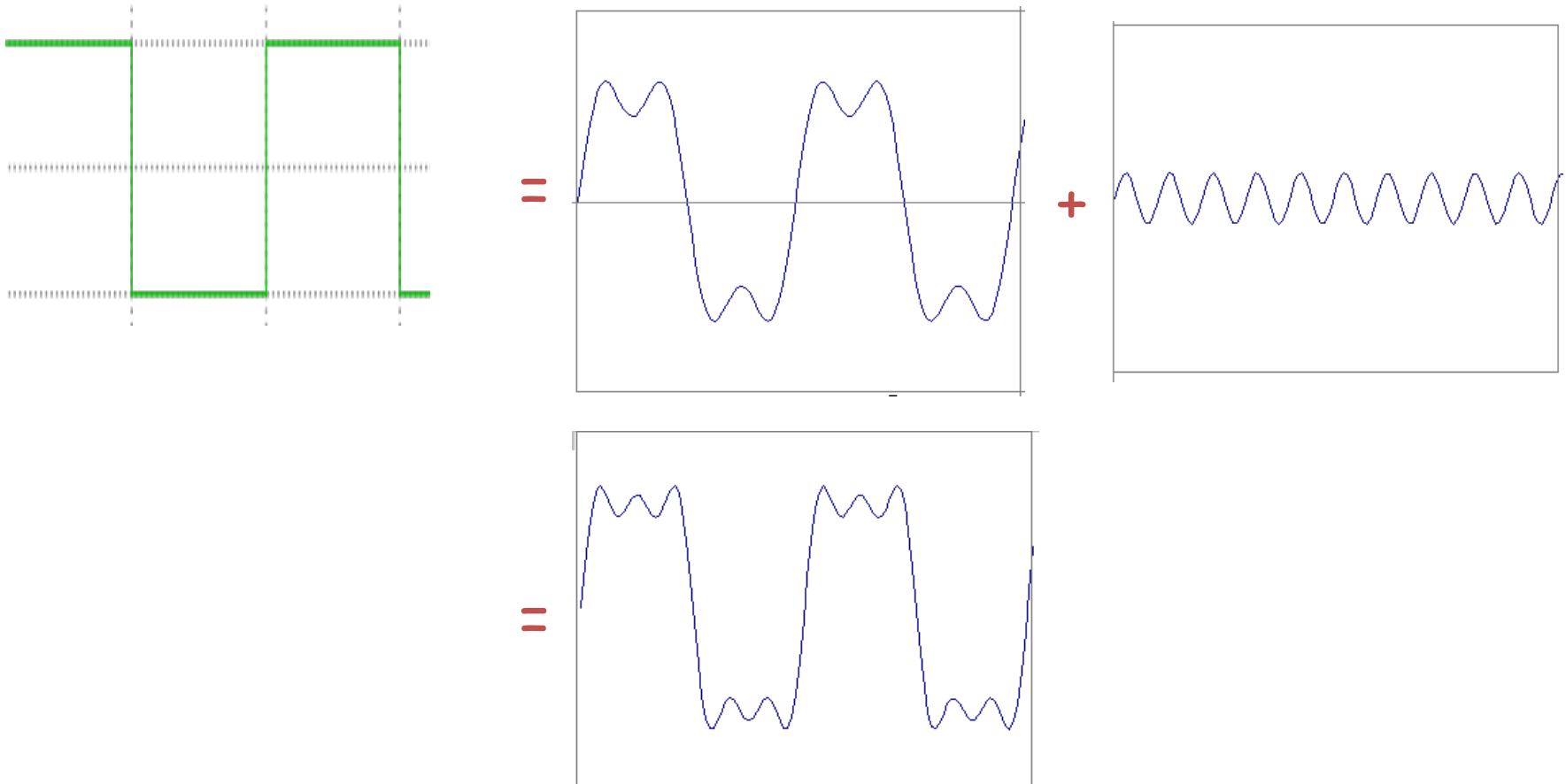
Frequency Spectra



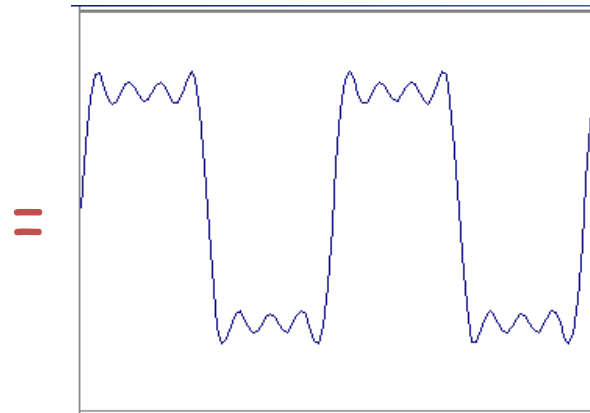
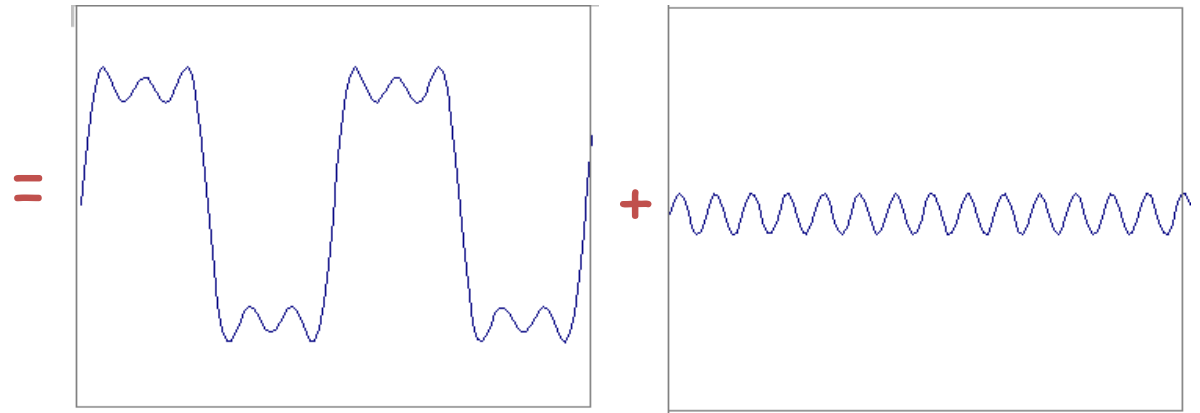
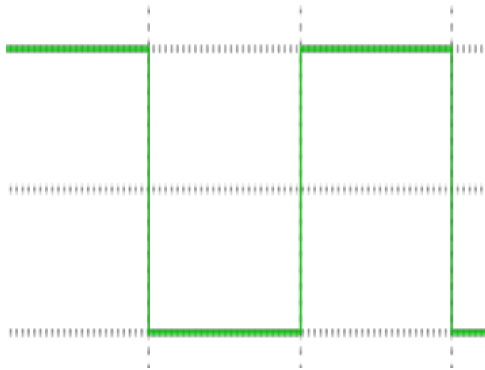
Frequency Spectra



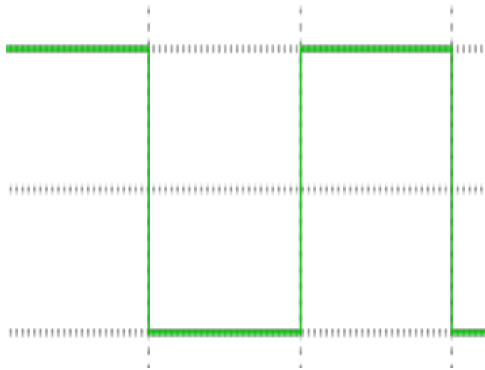
Frequency Spectra



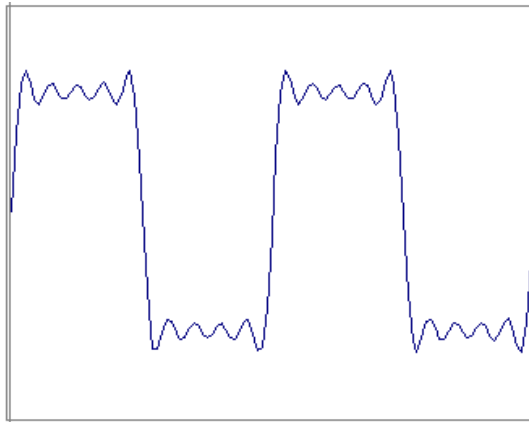
Frequency Spectra



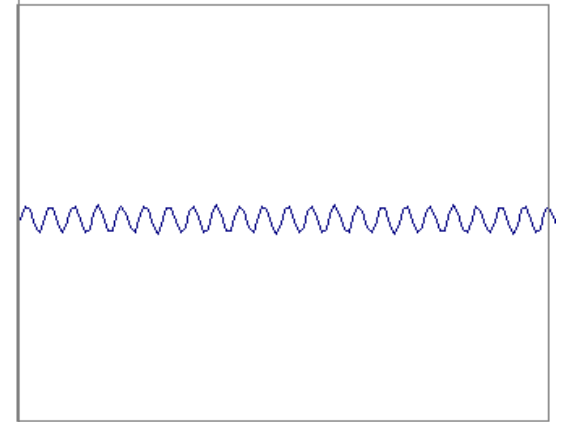
Frequency Spectra



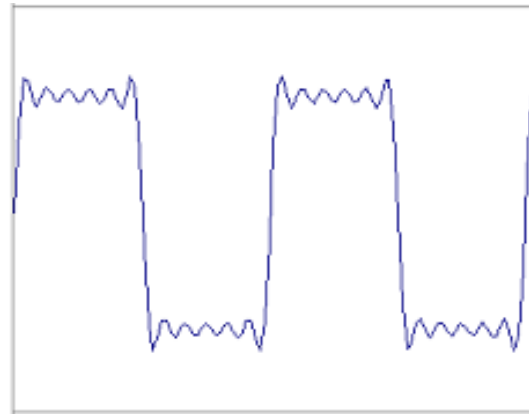
=



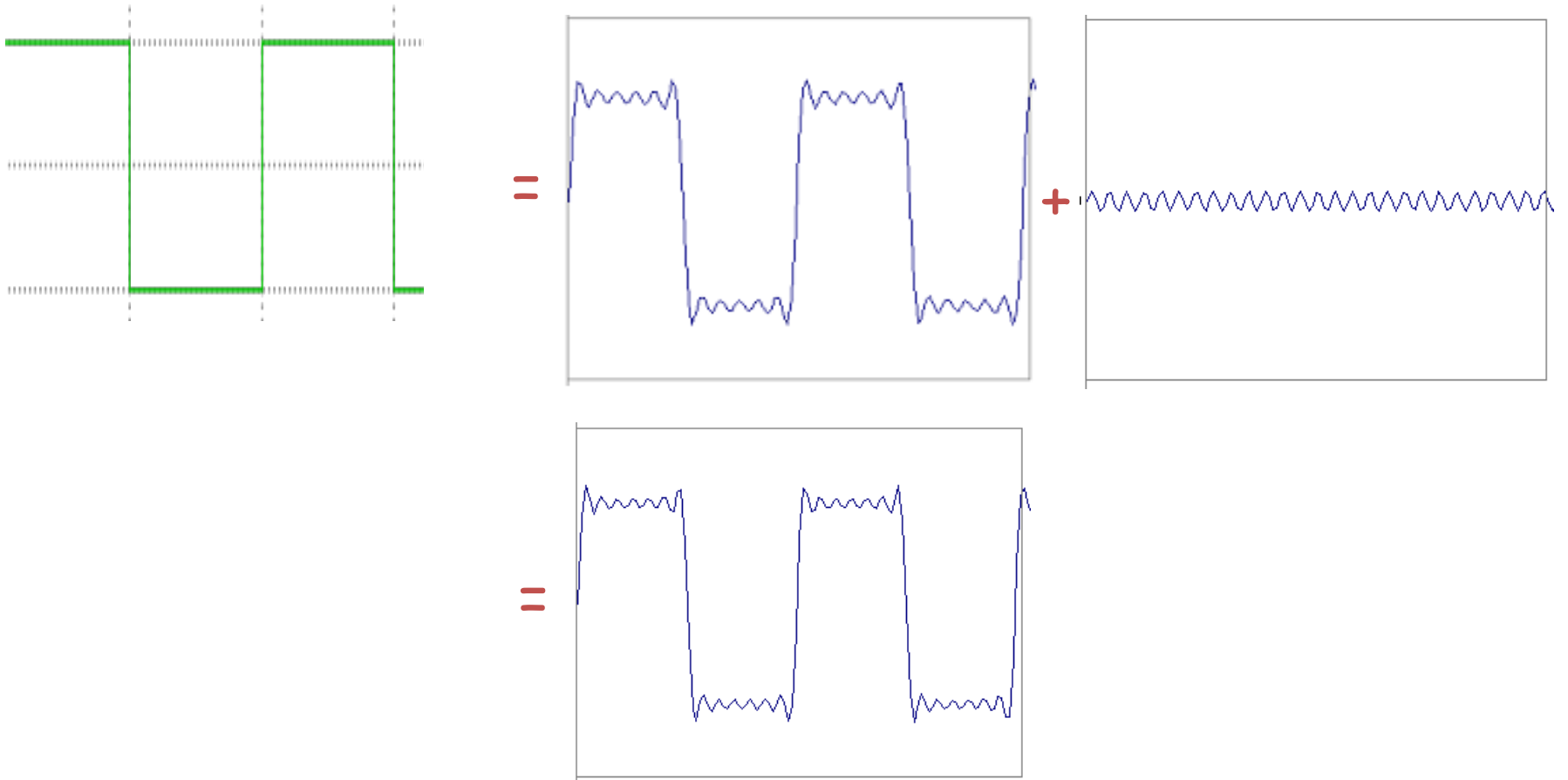
+



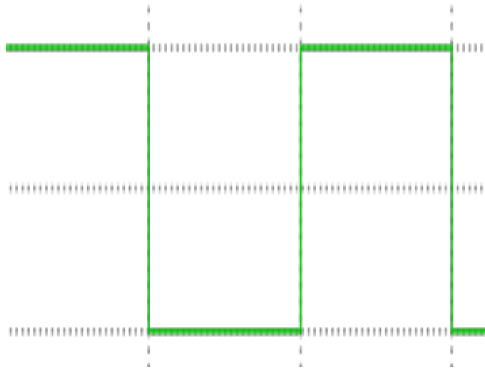
=



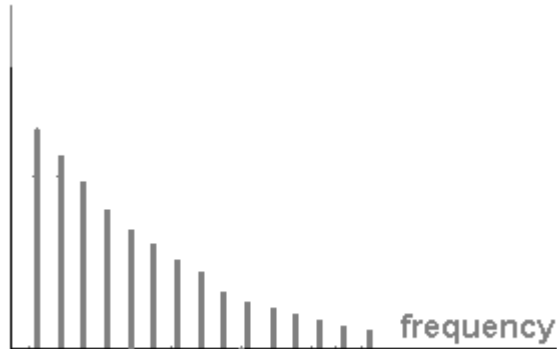
Frequency Spectra



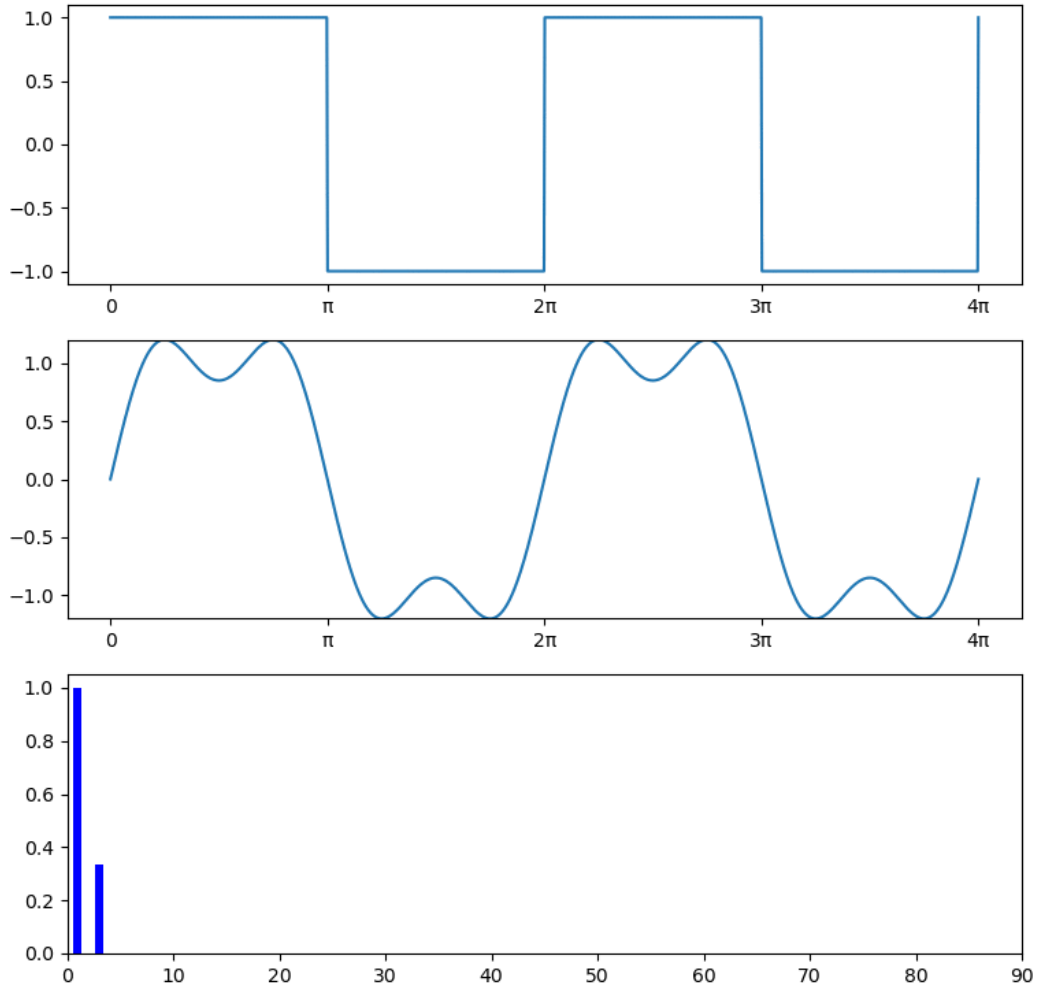
Frequency Spectra



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

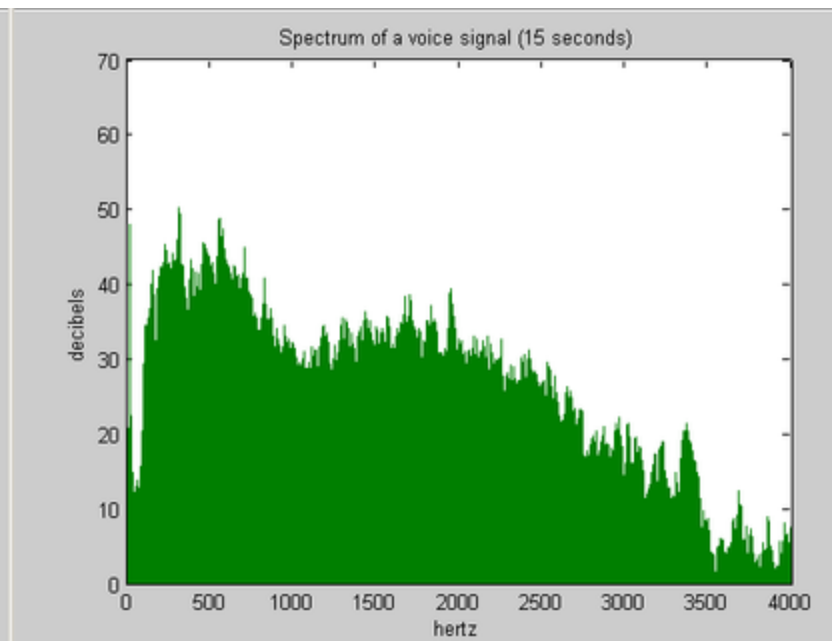
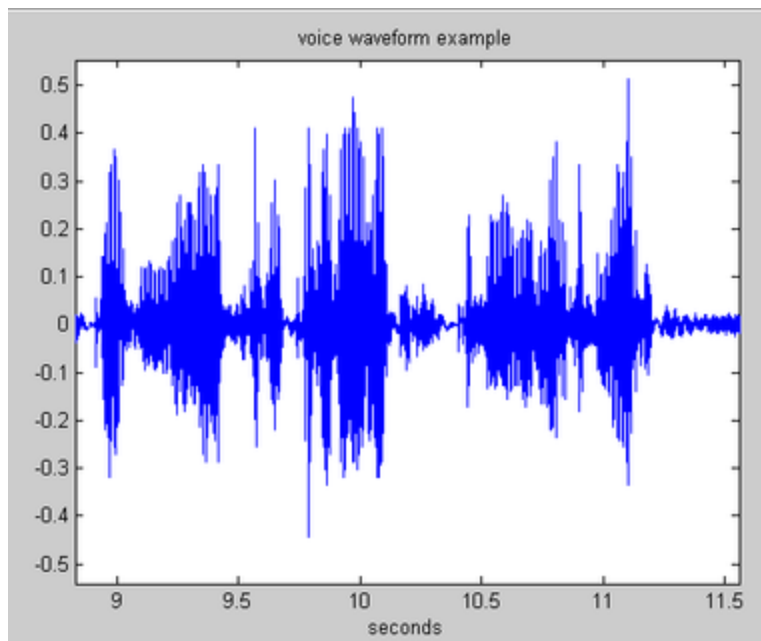


Frequency Spectra



Example: Music

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

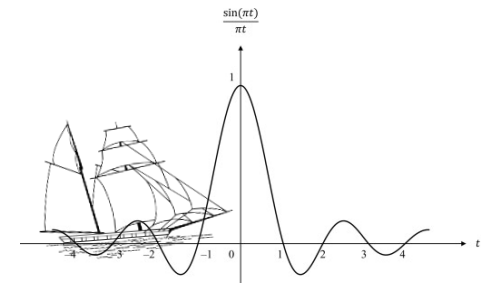


Other signals

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

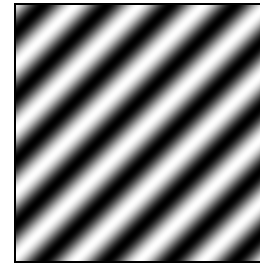
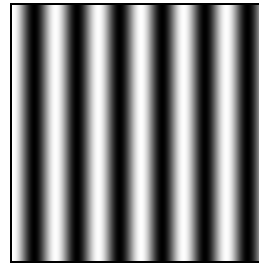
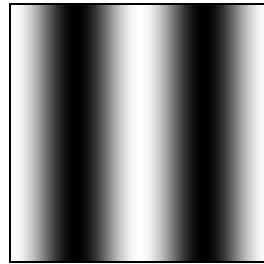
But don't take the Fourier transform of a square pulse while on a boat! *Why not?*

Because you don't wanna sinc!

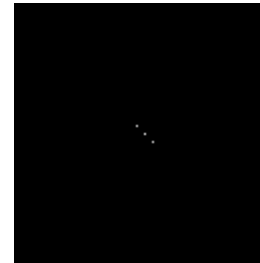
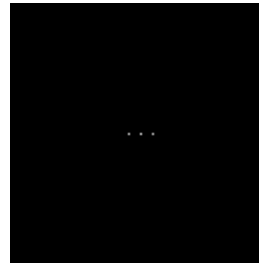
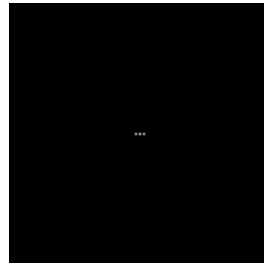


Fourier analysis in images

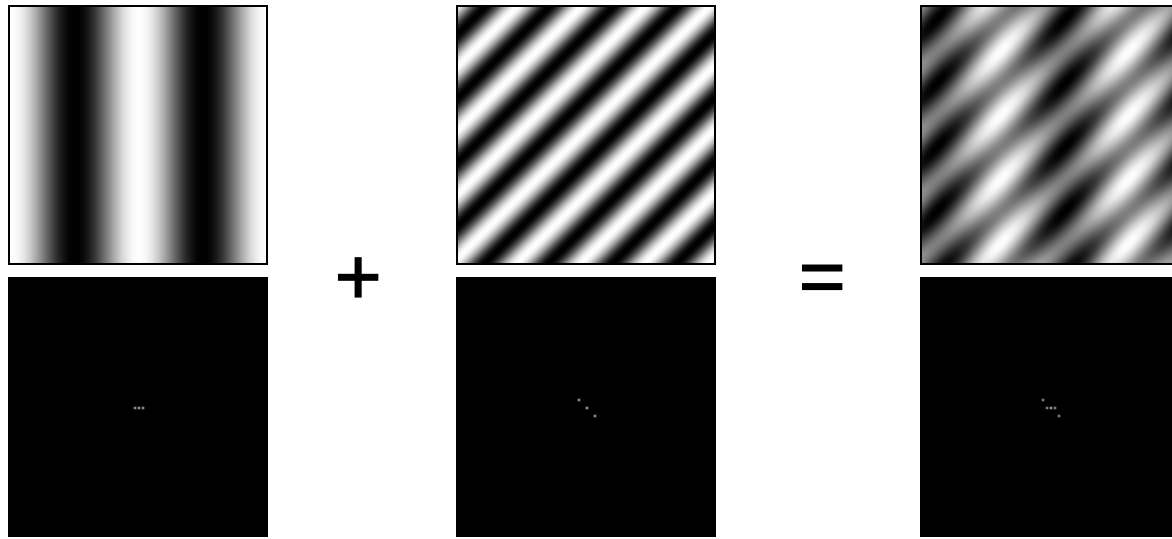
Intensity Image



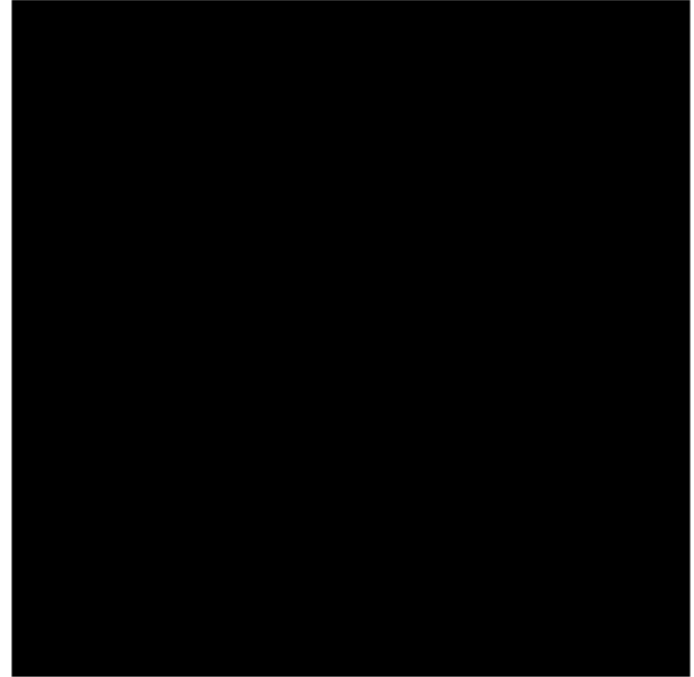
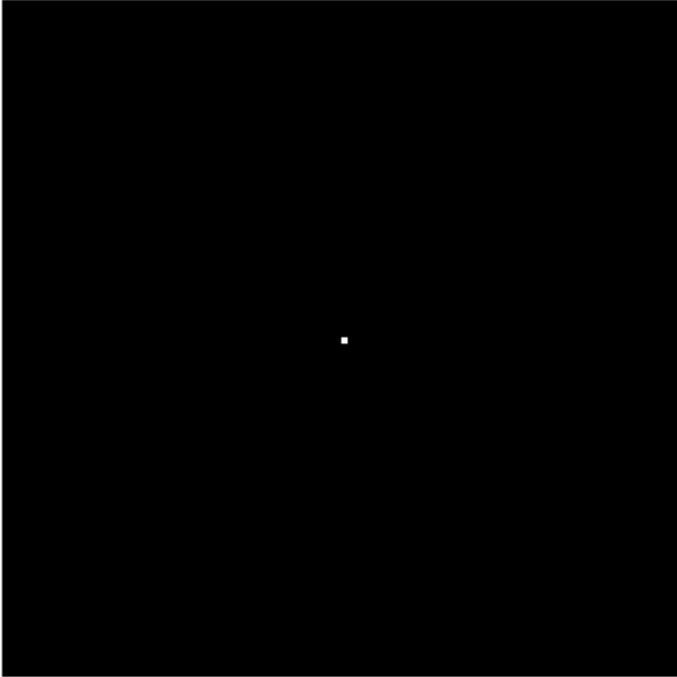
Fourier Image



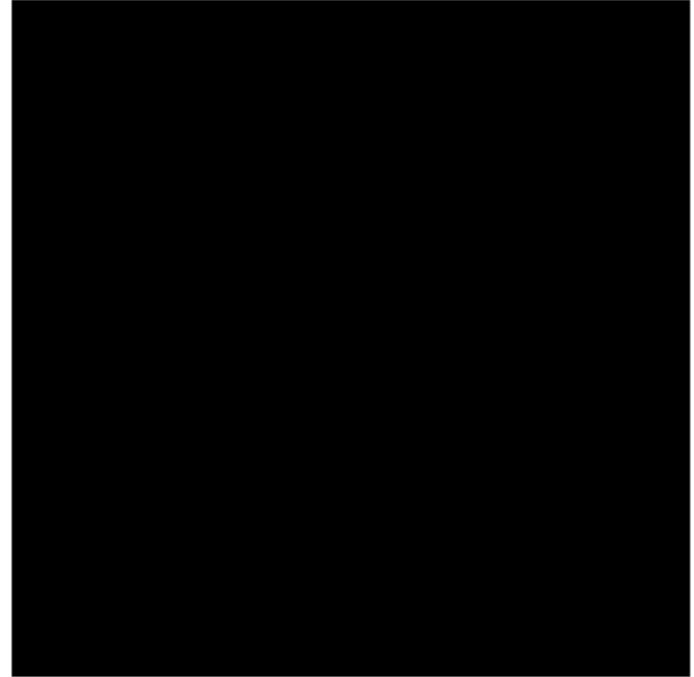
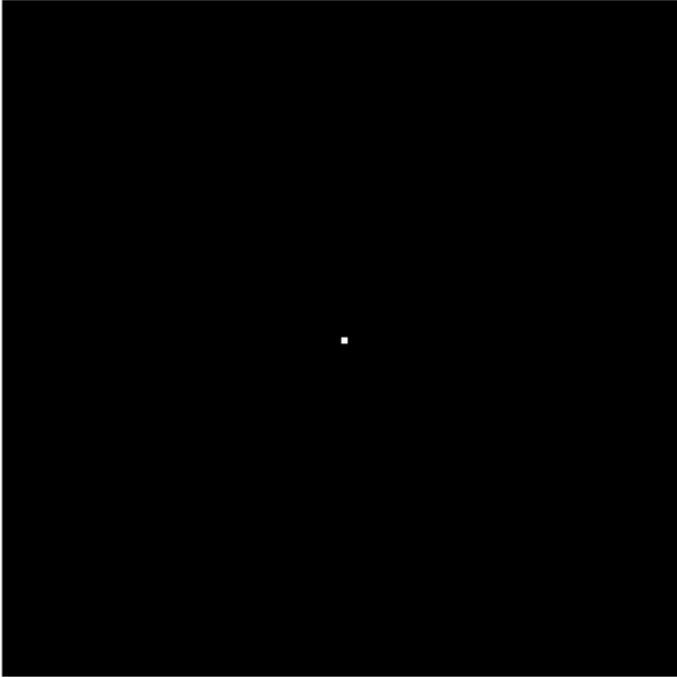
Signals can be composed



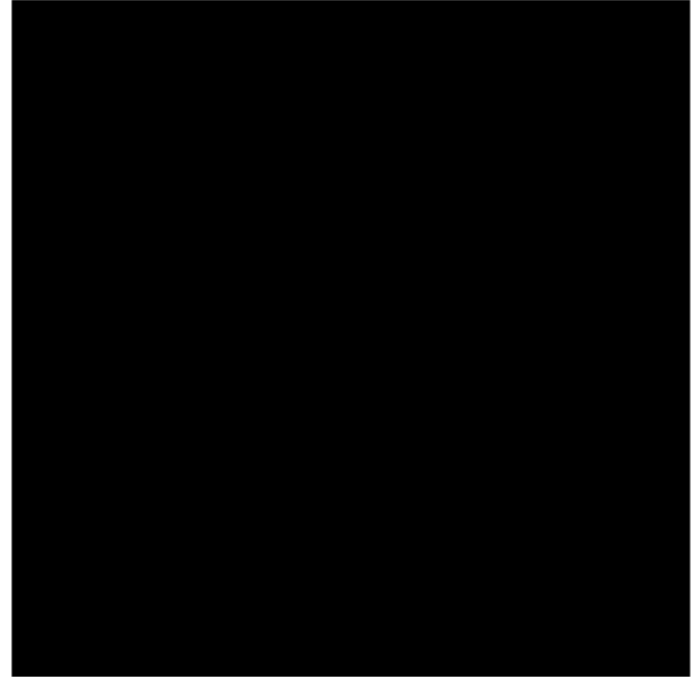
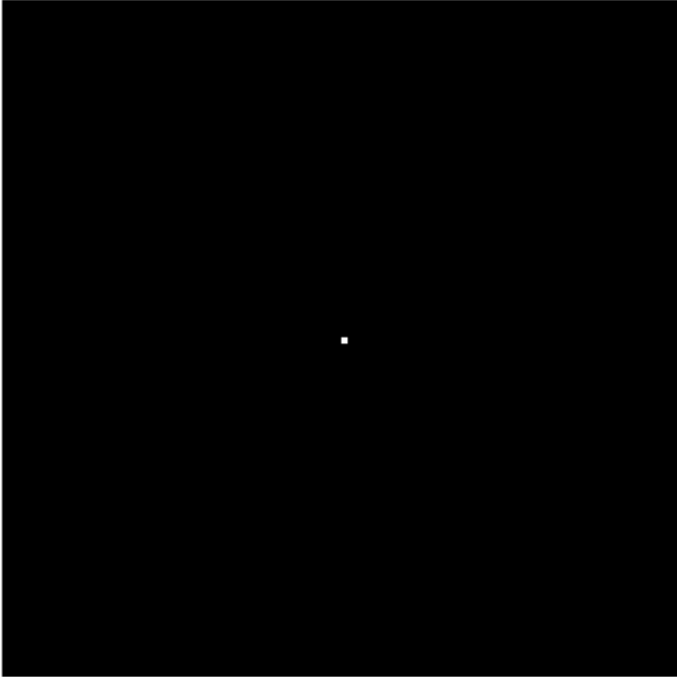
How change in frequency affects signal



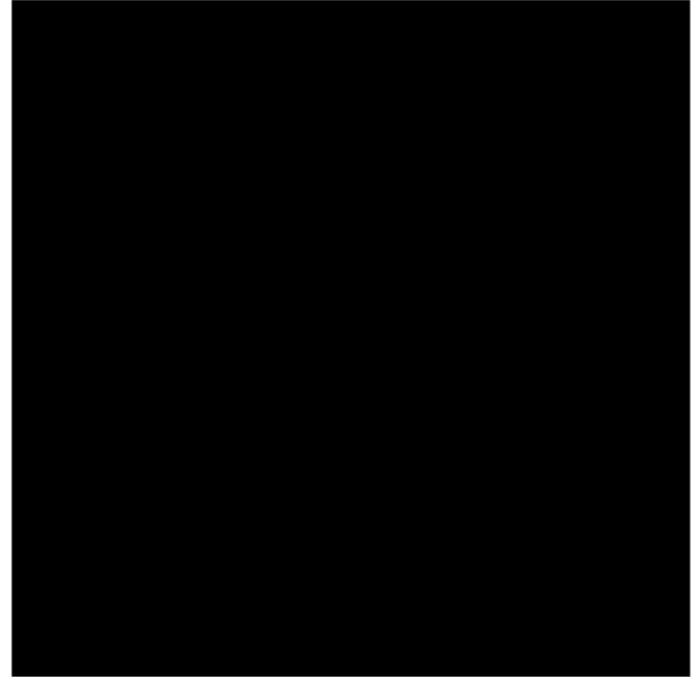
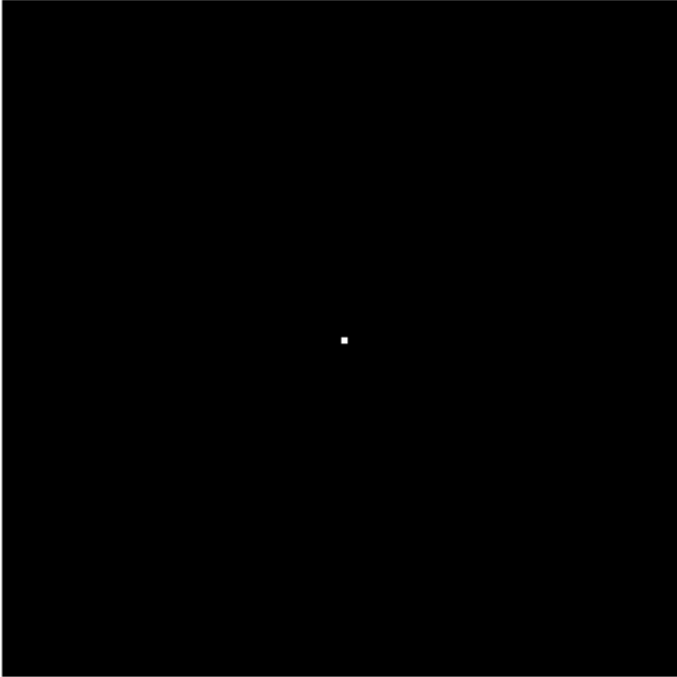
How change in frequency affects signal



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

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

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

Computing the Fourier Transform

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

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): $N \log N$

The Convolution Theorem

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

$$F[g * h] = F[g]F[h]$$

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

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

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

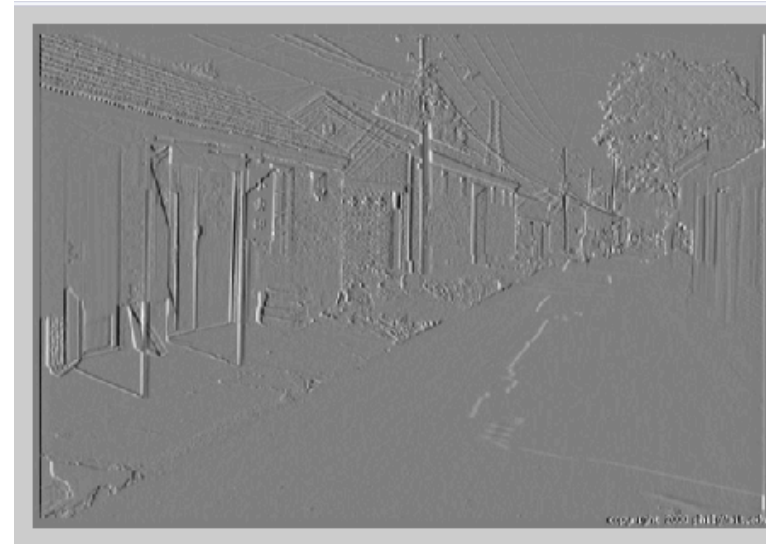
Properties of Fourier Transforms

- 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

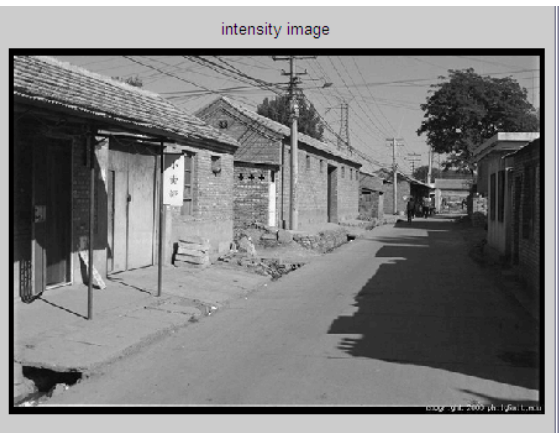
Filtering in spatial domain

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

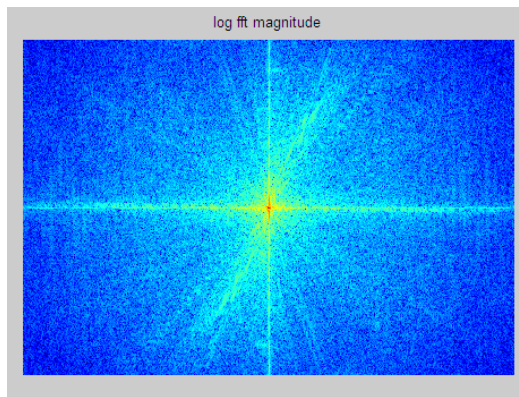
intensity image



Filtering in frequency domain



FFT

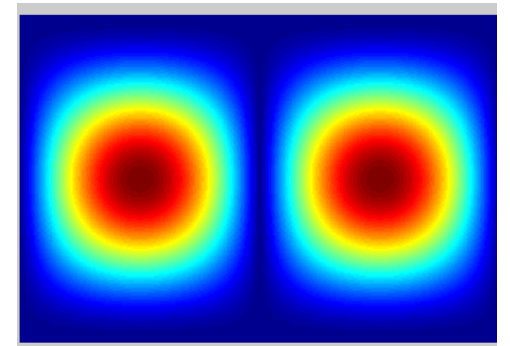


\times

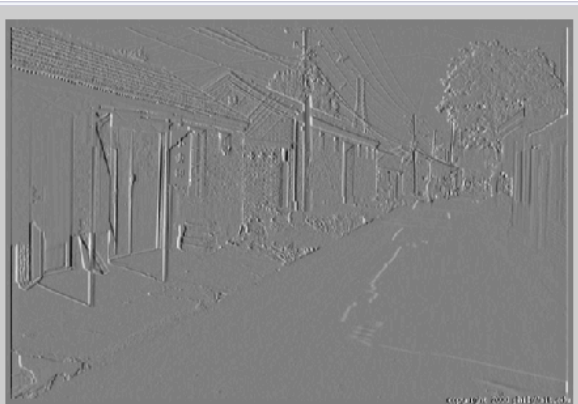
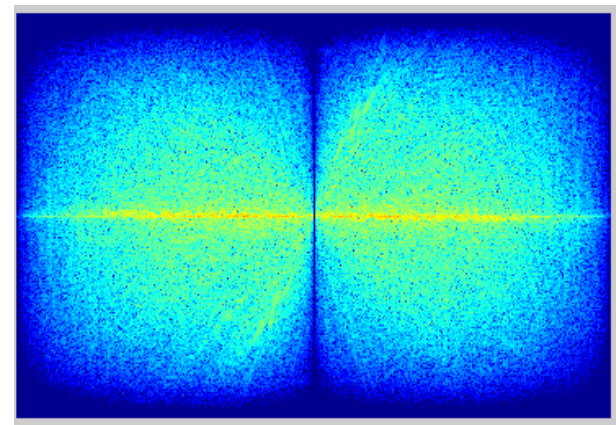
$=$



FFT



Inverse FFT

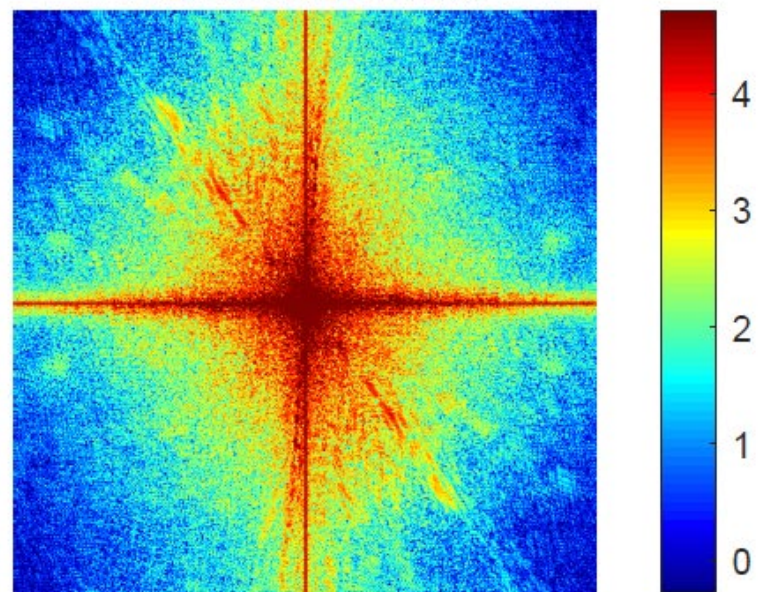


Fourier Image Examples

intensity image



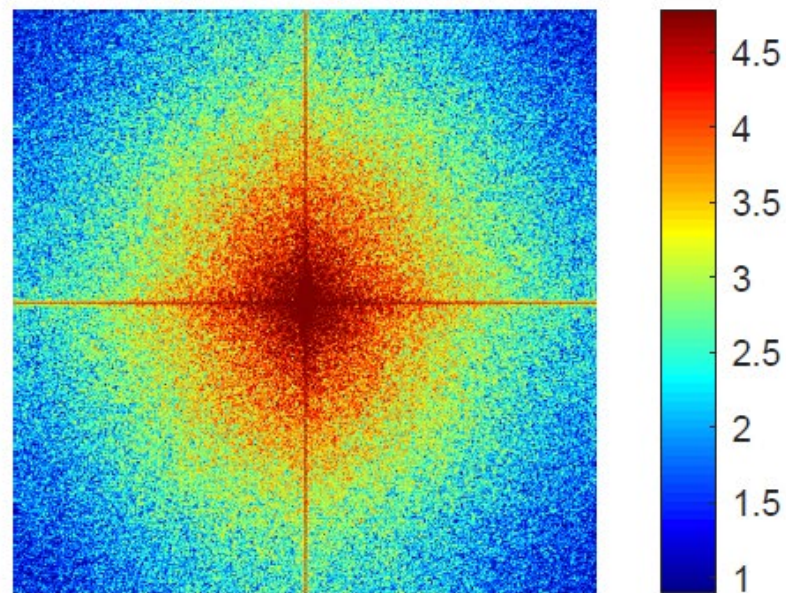
log fft magnitude



intensity image



log fft magnitude



intensity image



?

intensity image



?

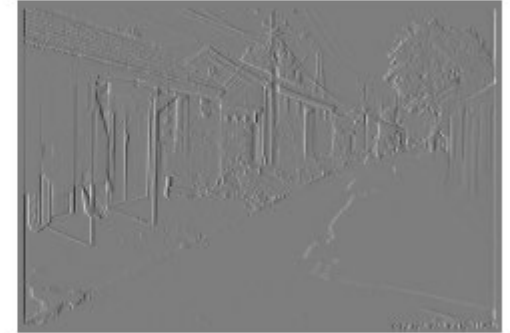
intensity image



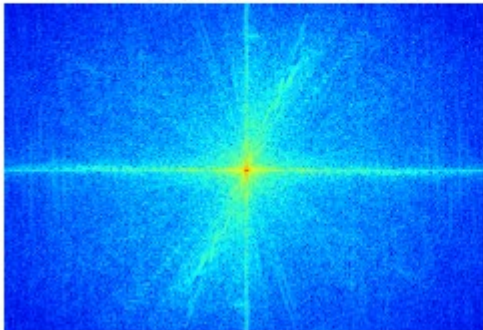
filter: sobel



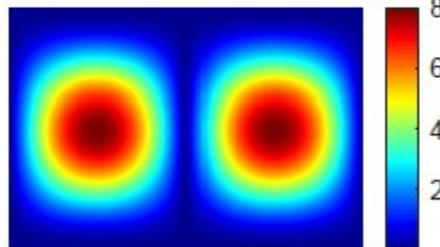
filtered image



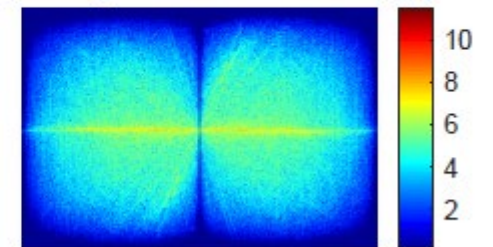
log fft magnitude of image



filter: sobel



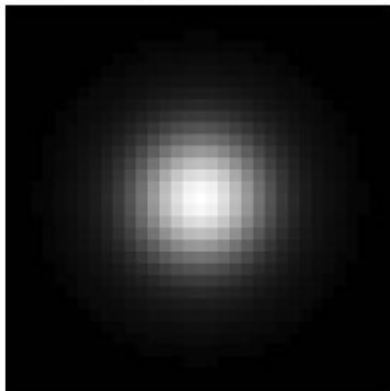
log fft magnitude of filtered image



intensity image



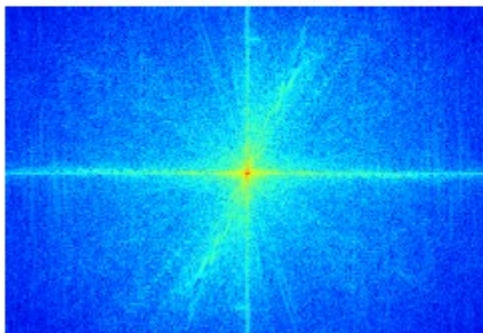
filter: gaussian



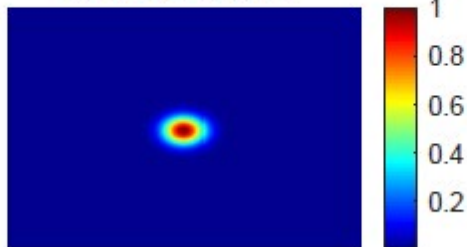
filtered image



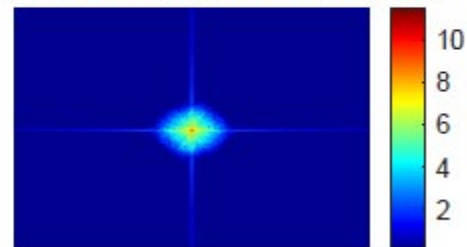
log fft magnitude of image



filter: gaussian



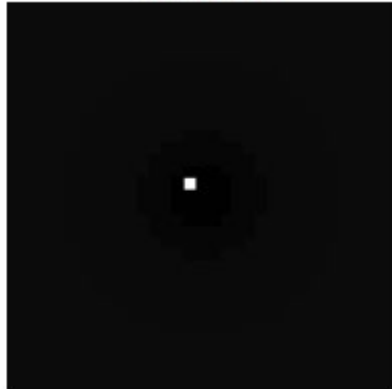
log fft magnitude of filtered image



intensity image



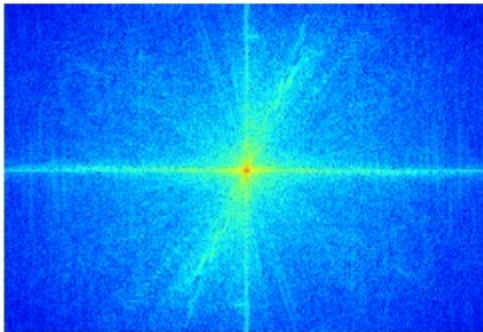
filter: log



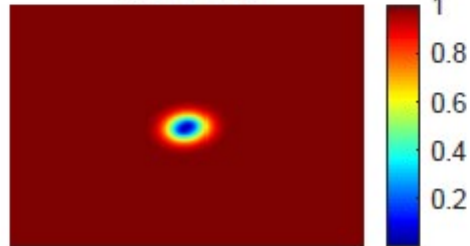
filtered image



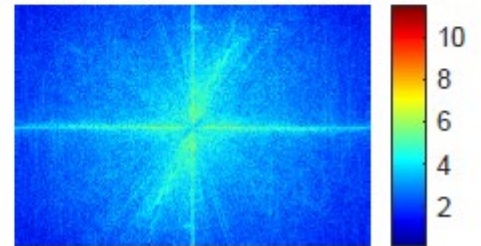
log fft magnitude of image



filter: log



log fft magnitude of filtered image



intensity image



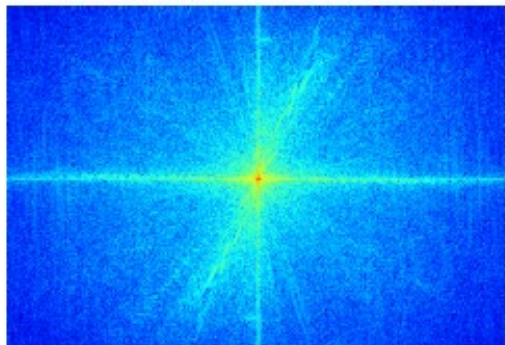
filter: box



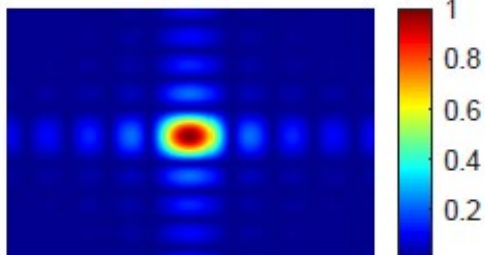
filtered image



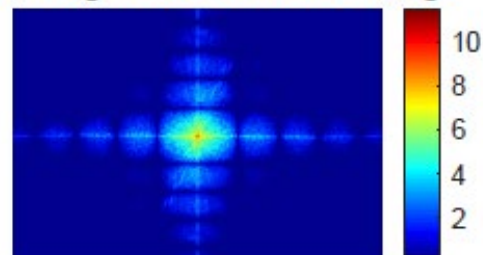
log fft magnitude of image



filter: box



log fft magnitude of filtered image



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?

```
% (Matlab code)
%% 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);

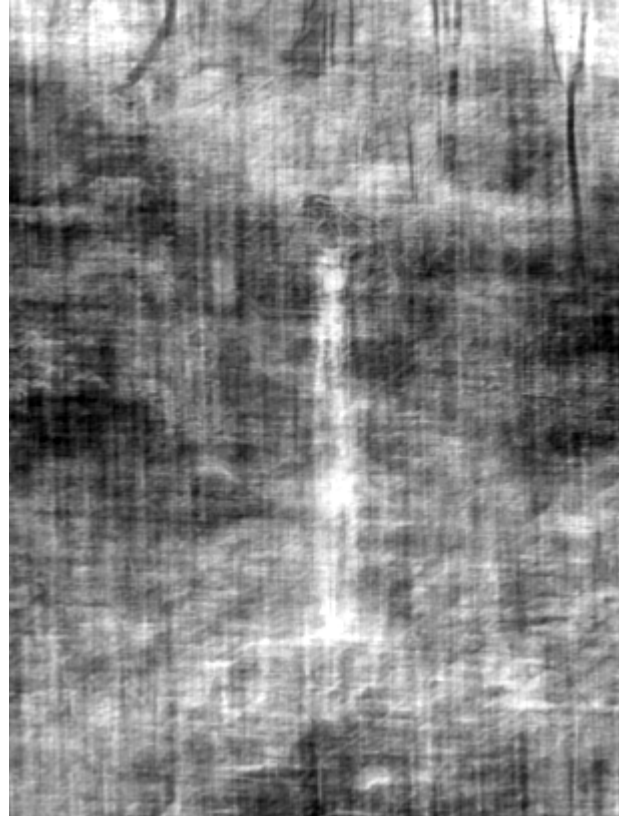
%% Combine mag and phase from different images and compute inverse FFT
mag1_phase2 = ifft2(im1_fft_mag.*cos(im2_fft_phase)+1i*im1_fft_mag.*sin(im2_fft_phase));
phase1_mag2 =ifft2(im2_fft_mag.*cos(im1_fft_phase)+1i*im2_fft_mag.*sin(im1_fft_phase));
```



Amplitude

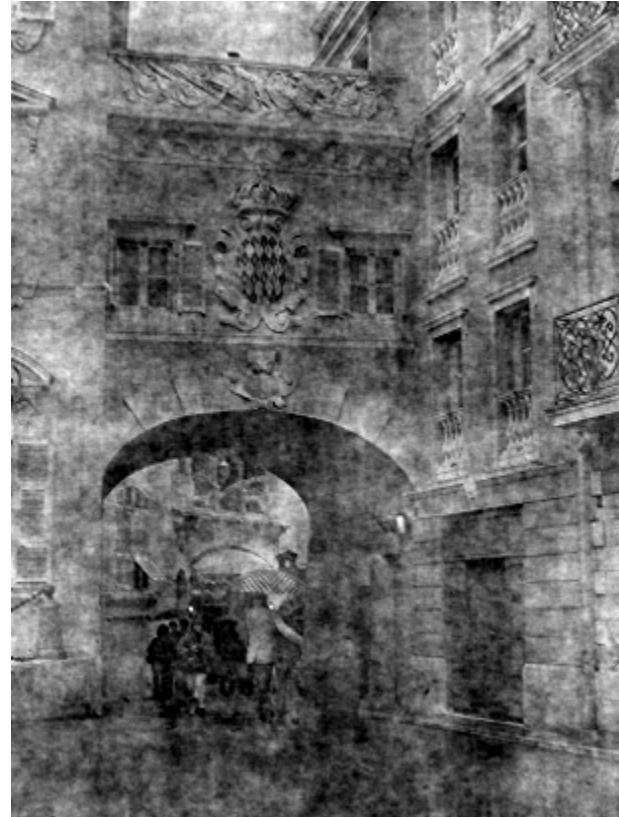


Phase





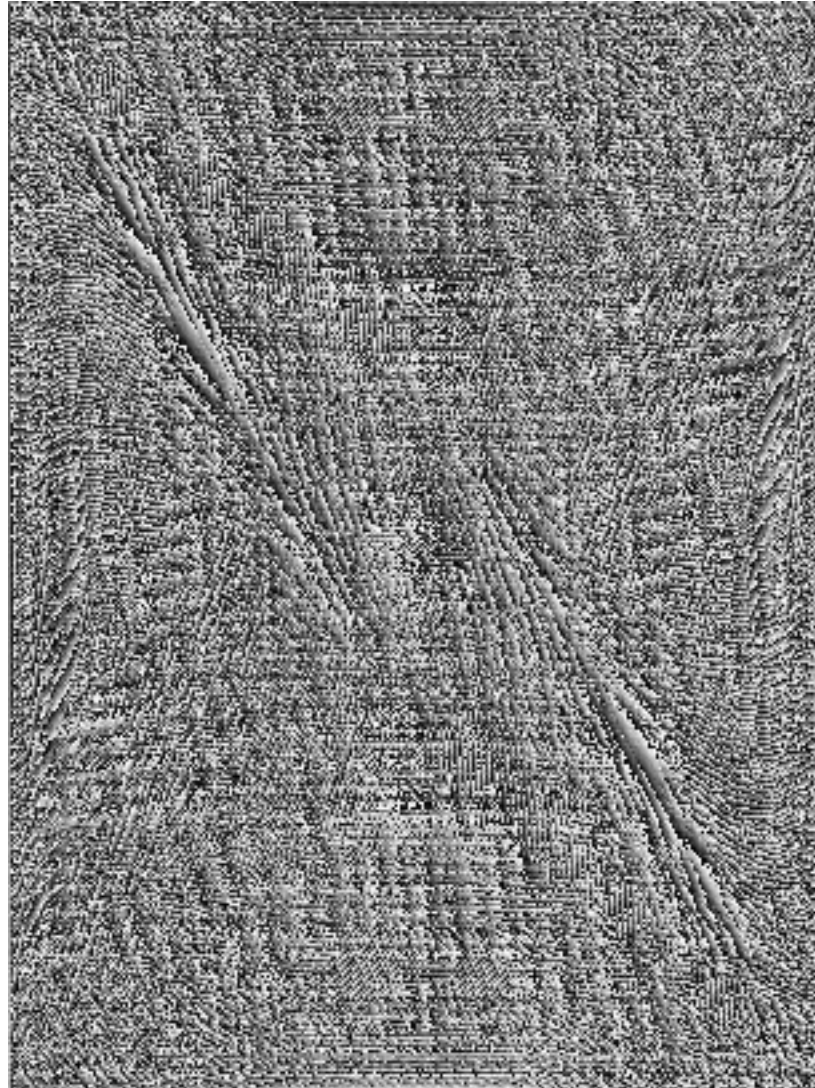
Phase



Amplitude



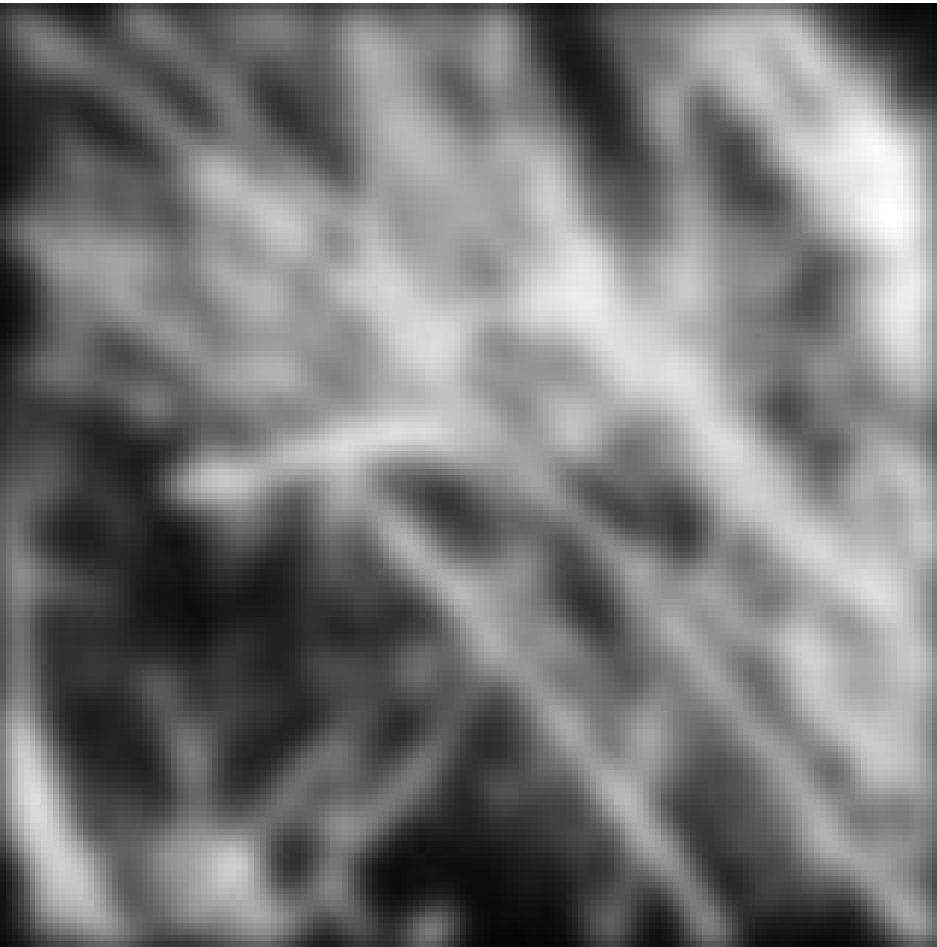
Phase Image



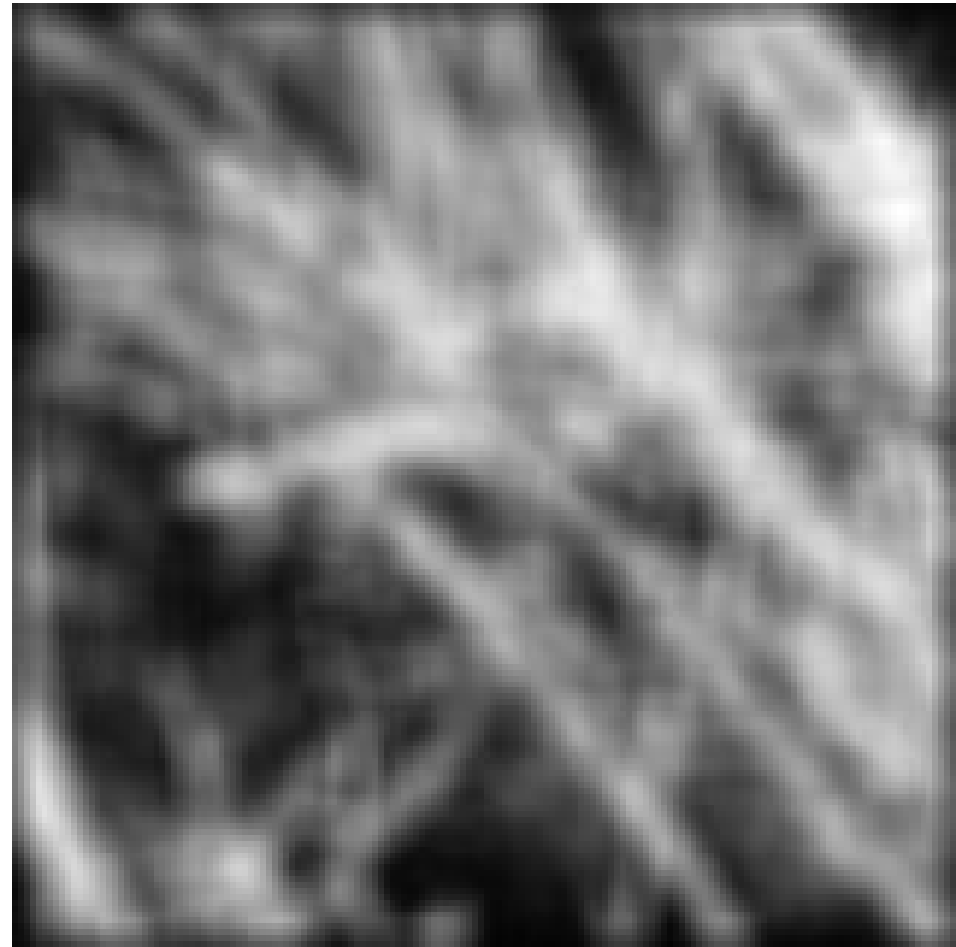
Filtering

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

Gaussian



Box filter

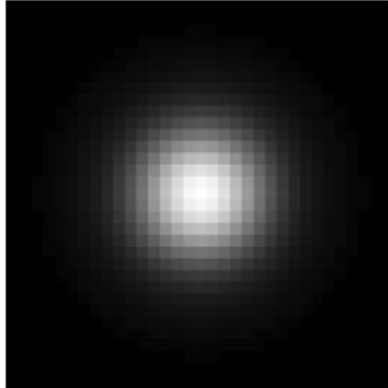


Gaussian filter

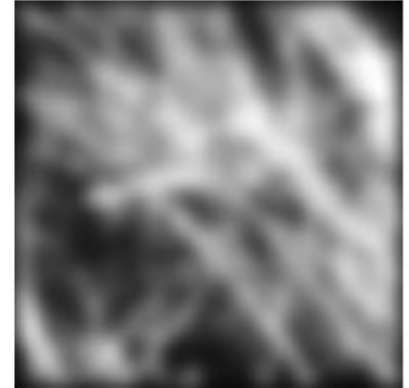
intensity image



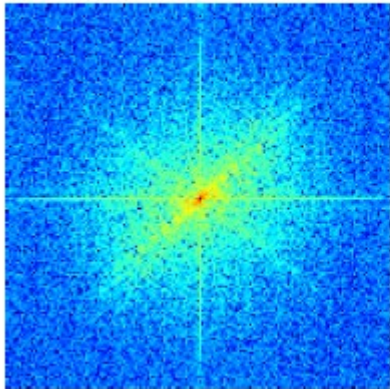
filter: gaussian



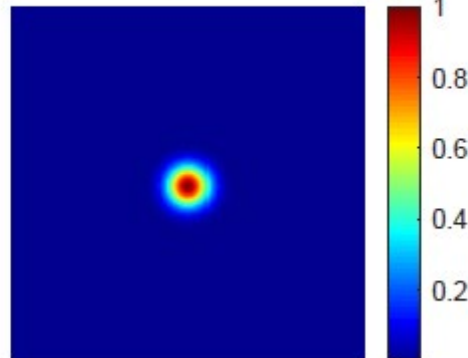
filtered image



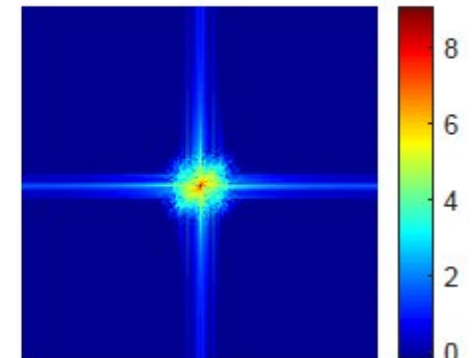
log fft magnitude of image



filter: gaussian



log fft magnitude of filtered image



Box filter

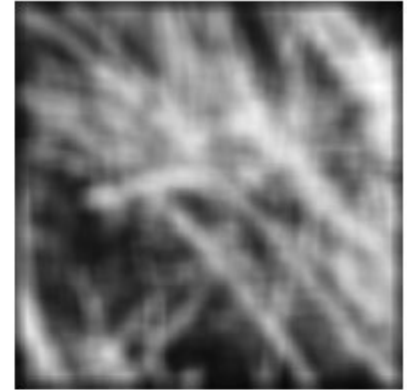
intensity image



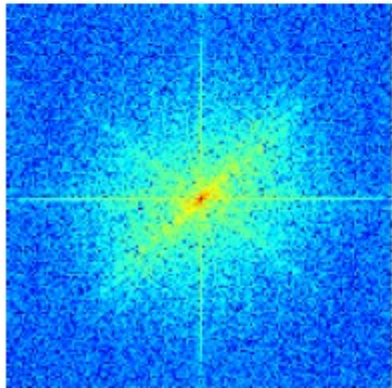
filter: box



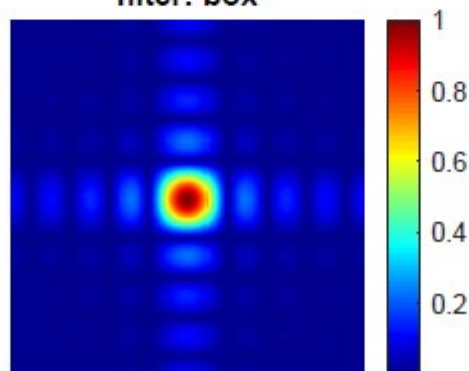
filtered image



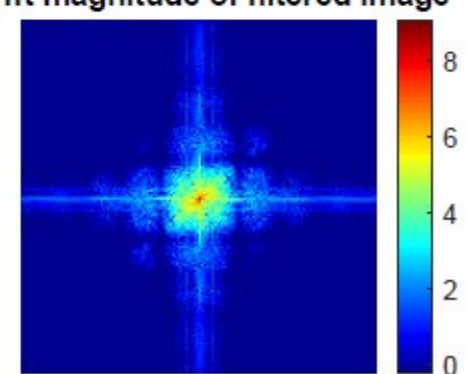
log fft magnitude of image



filter: box

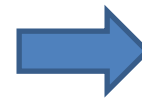


log fft magnitude of filtered image



Sampling

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



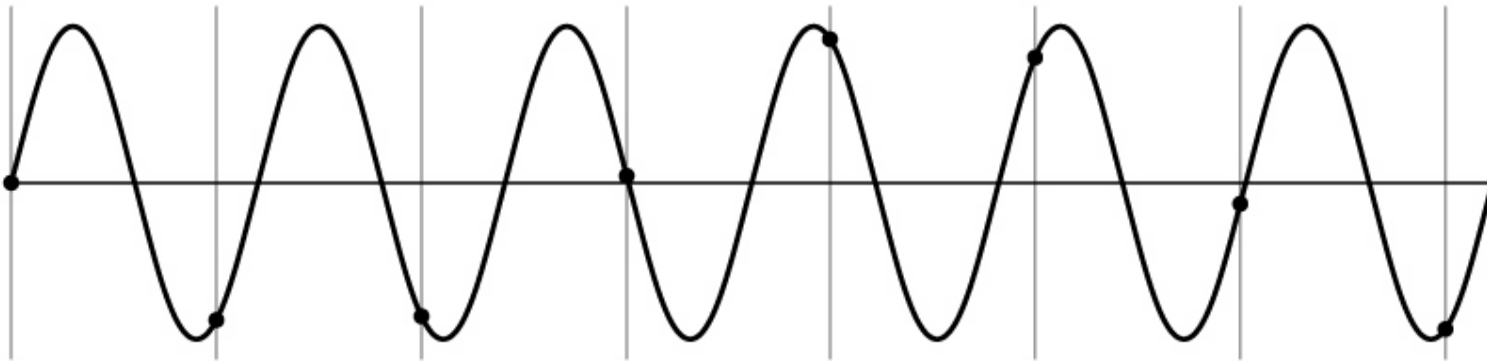
Subsampling by a factor of 2



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

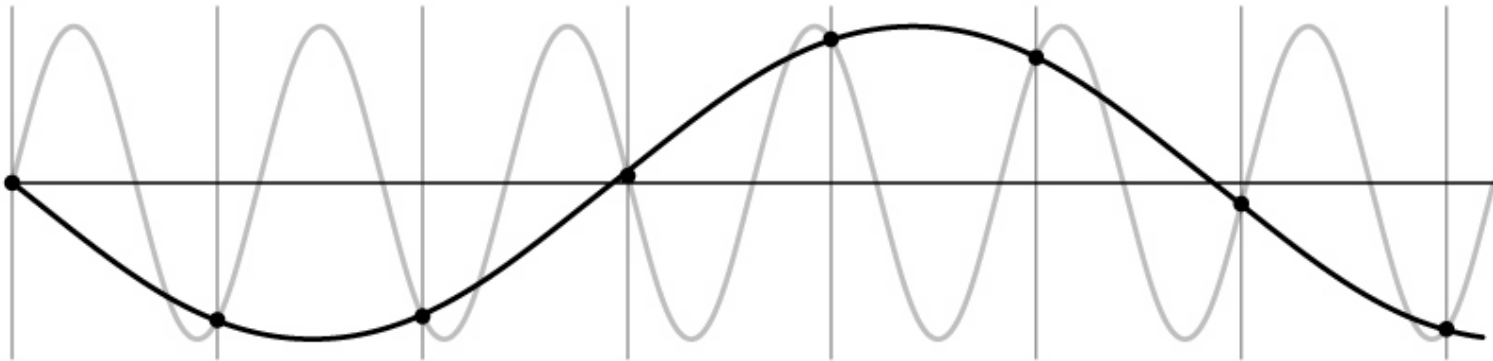
Aliasing problem

- 1D example (sinewave):



Aliasing problem

- 1D example (sinewave):



Aliasing problem

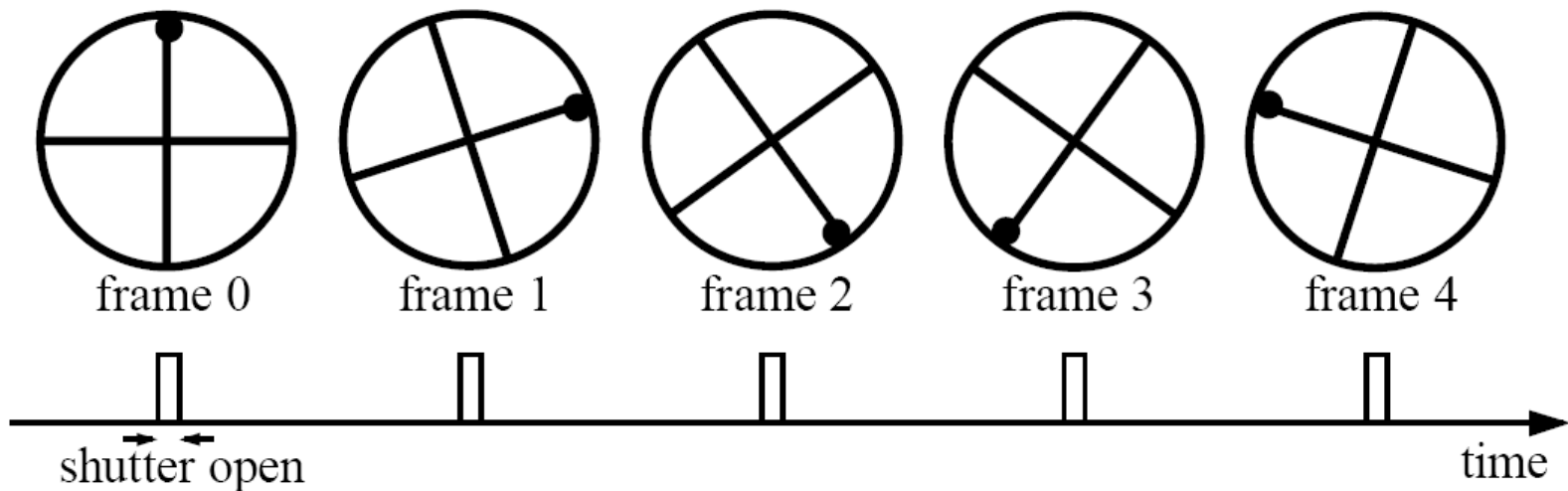
- Sub-sampling may be dangerous....
- Characteristic errors may appear:
 - “Wagon wheels rolling the wrong way in movies”
 - “Checkerboards disintegrate in ray tracing”
 - “Striped shirts look funny on color television”

Aliasing in video

Imagine a spoked wheel moving to the right (rotating clockwise).

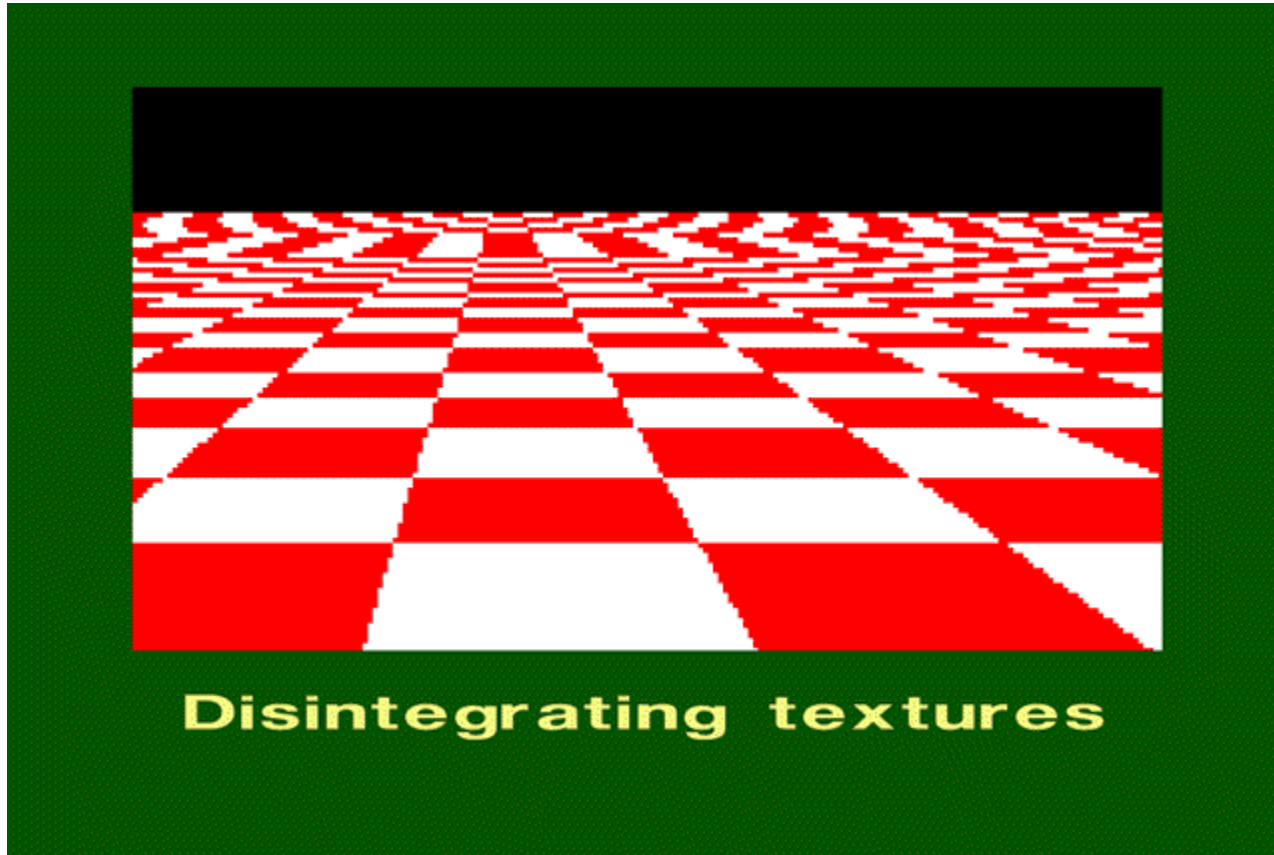
Mark wheel with dot so we can see what's happening.

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)

Aliasing in graphics



Sampling and aliasing

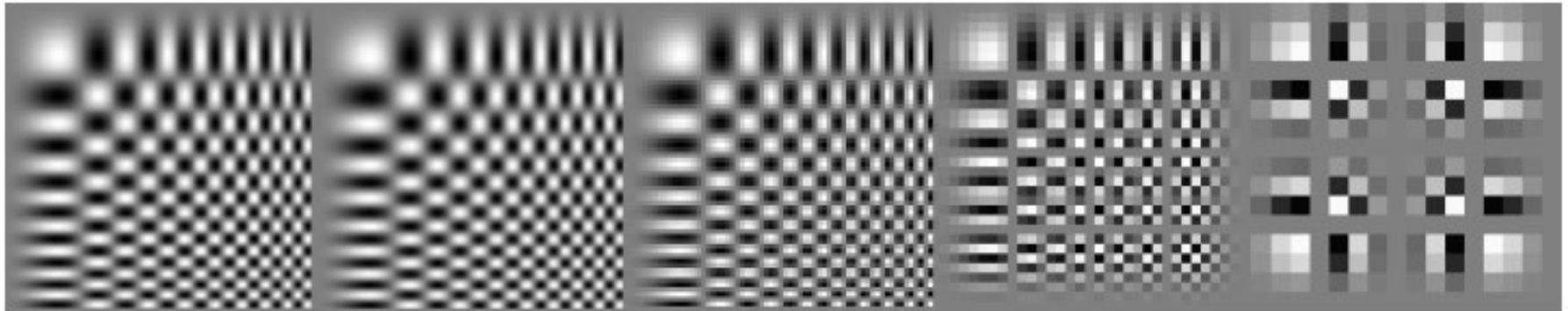
256x256

128x128

64x64

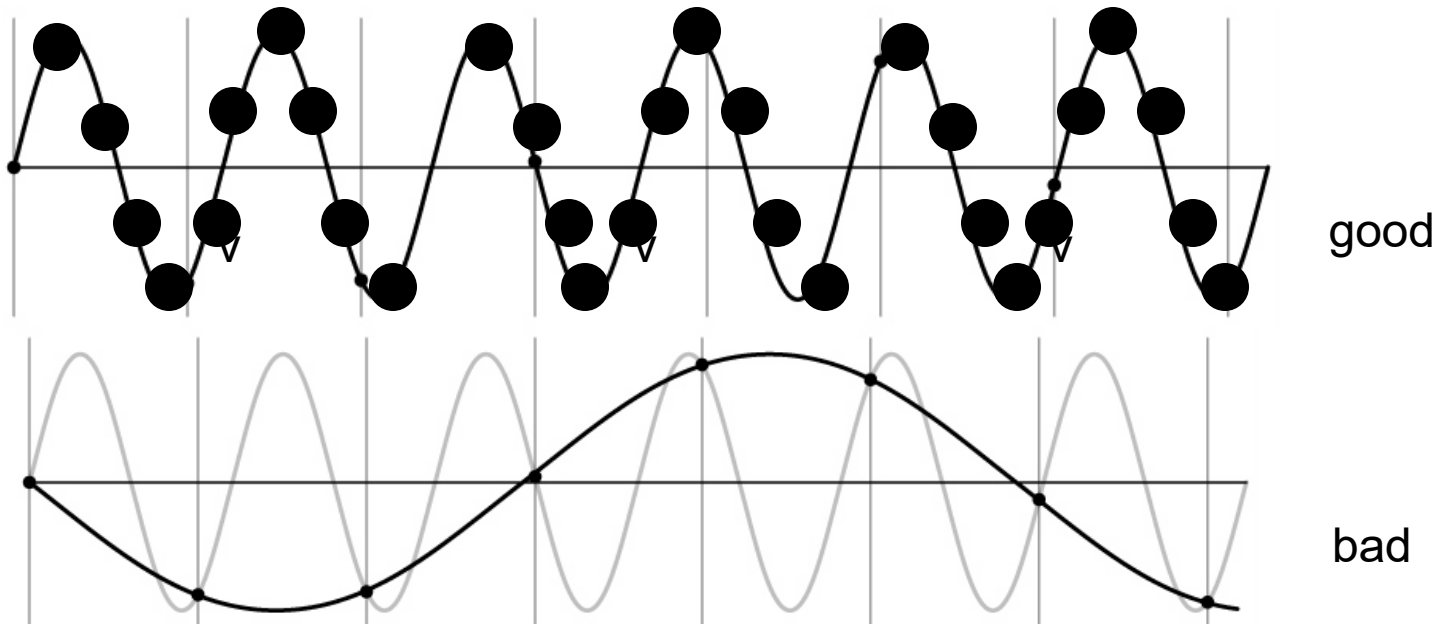
32x32

16x16



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 allow 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

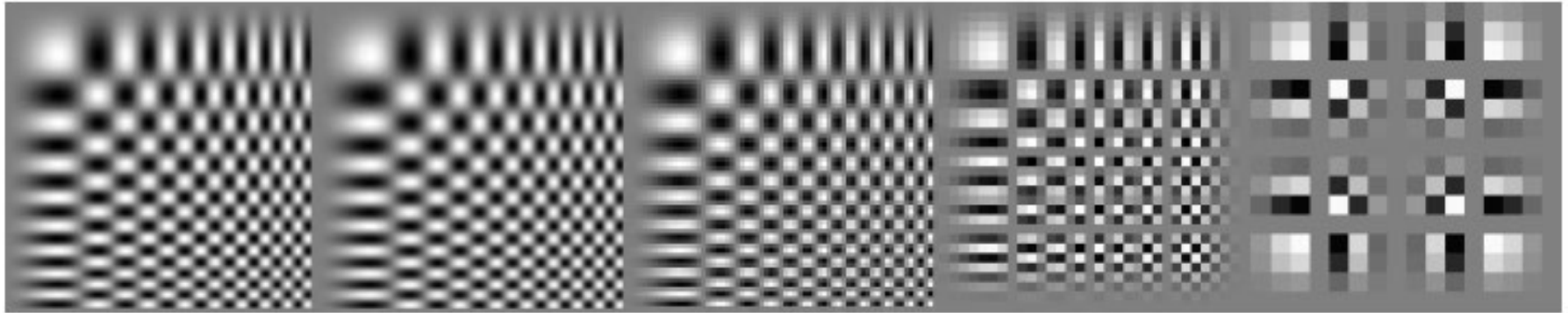
256x256

128x128

64x64

32x32

16x16



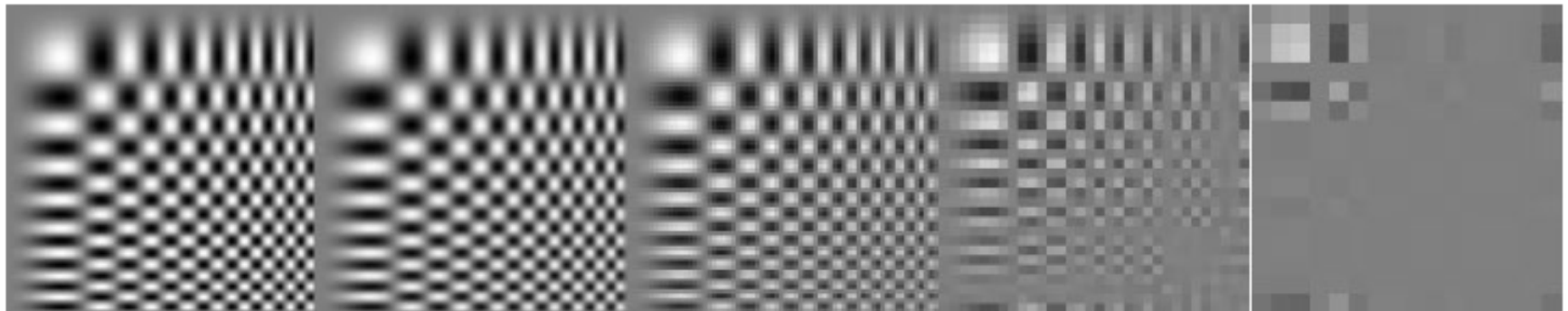
256x256

128x128

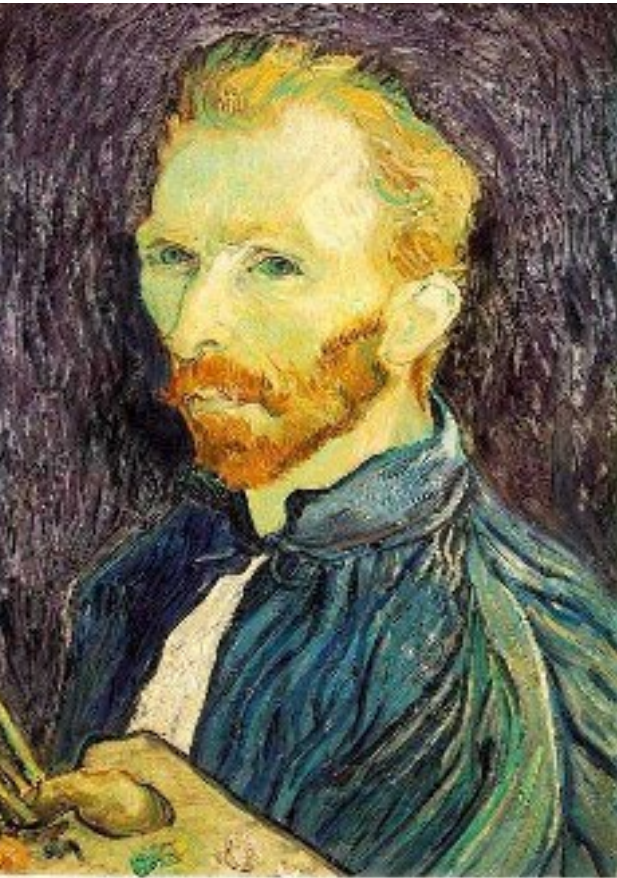
64x64

32x32

16x16



Subsampling without pre-filtering



1/2



1/4 (2x zoom)

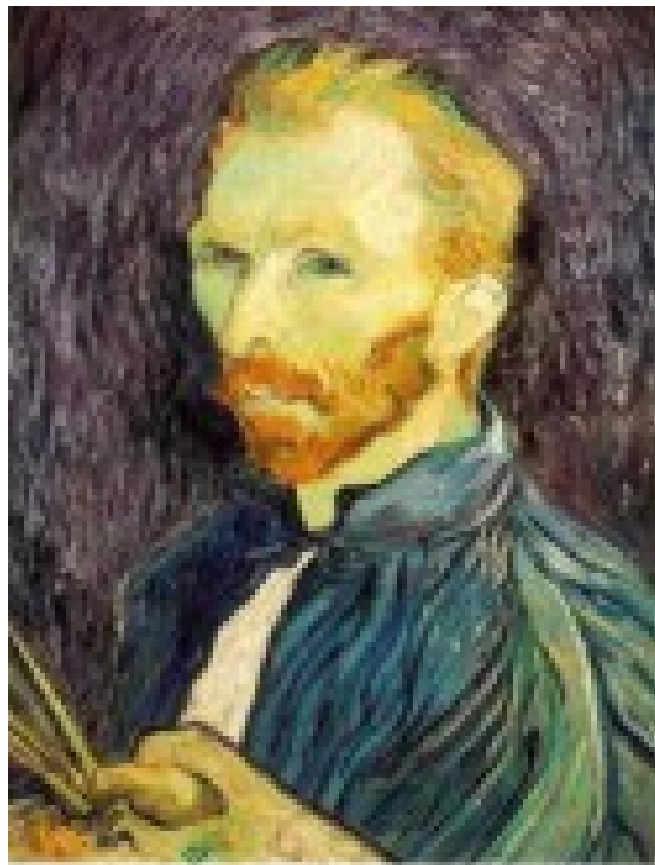


1/8 (4x zoom)

Subsampling with Gaussian pre-filtering



Gaussian $1/2$

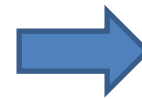


G $1/4$

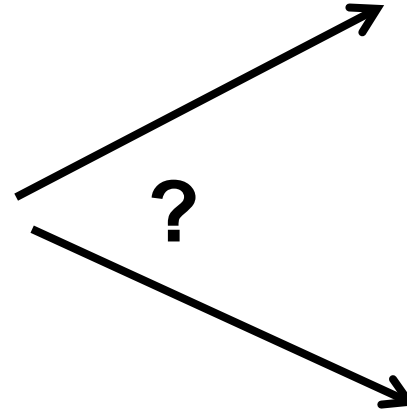


G $1/8$

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

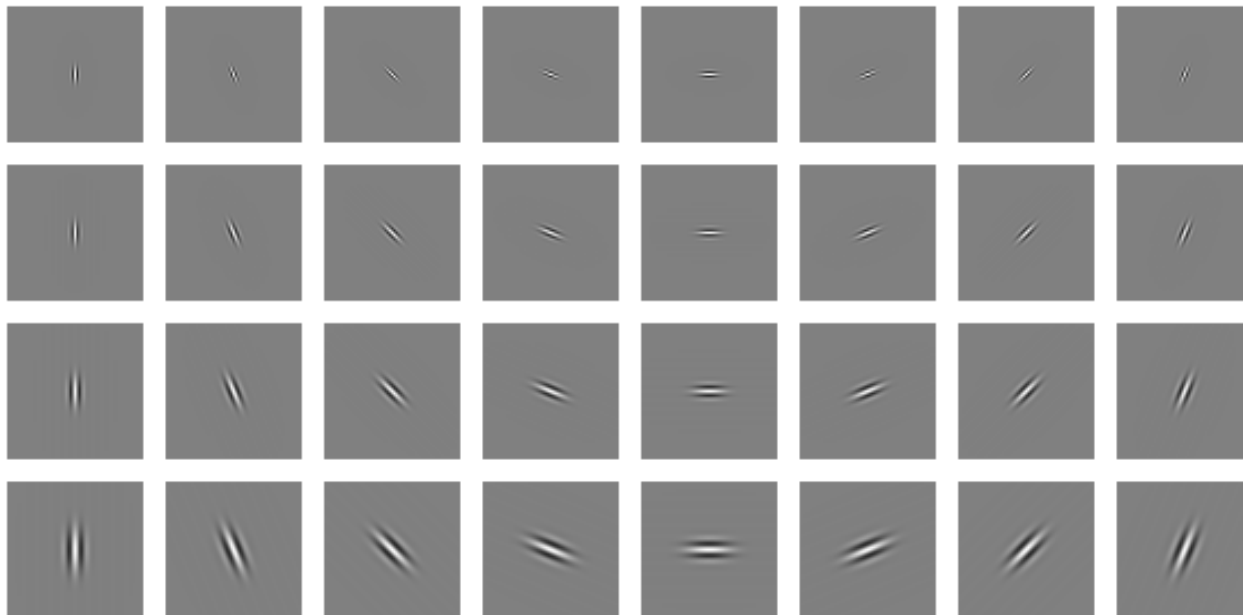


Why do we get different, distance-dependent interpretations of hybrid images?



Clues from Human Perception

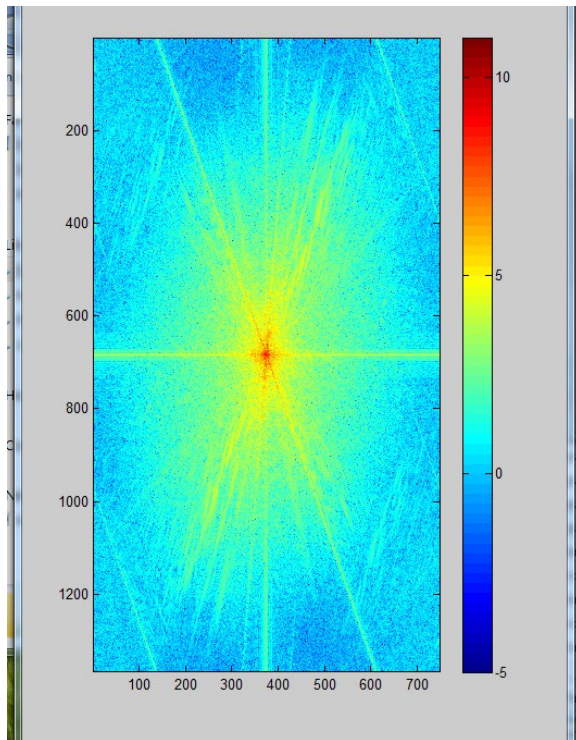
- Early processing in humans filters for various orientations and scales of frequency
- Perceptual cues in the mid frequencies dominate perception
- When we see an image from far away, we are effectively subsampling it



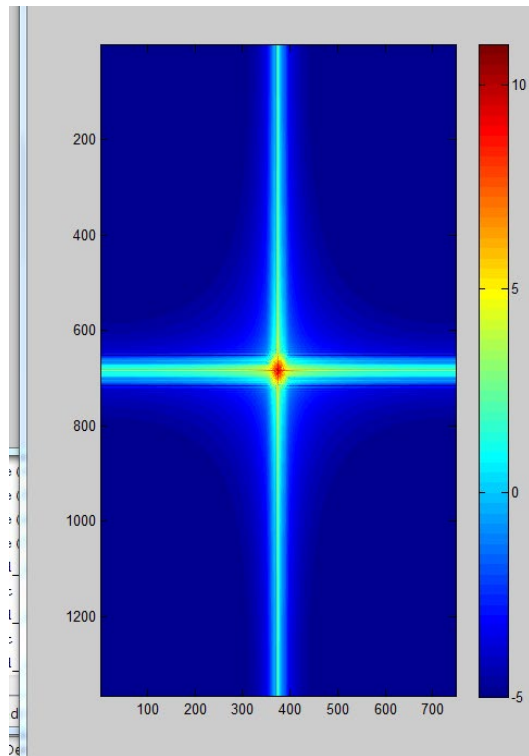
Early Visual Processing: Multi-scale edge and blob filters

Hybrid Image in FFT

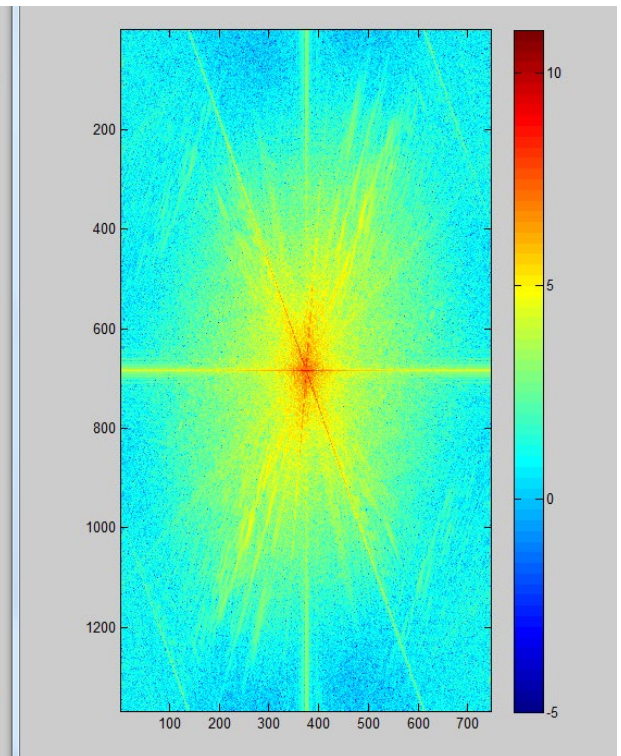
Hybrid Image



Low-passed Image

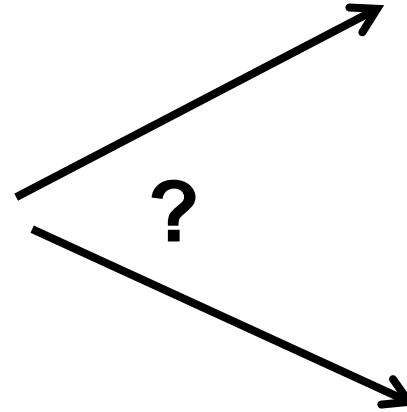


High-passed Image



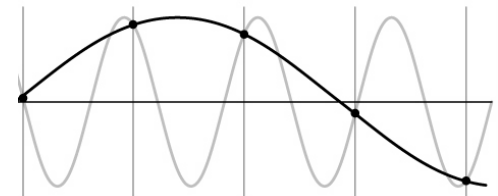
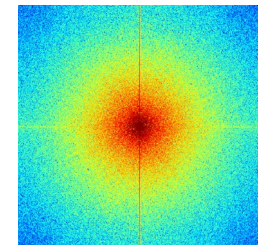
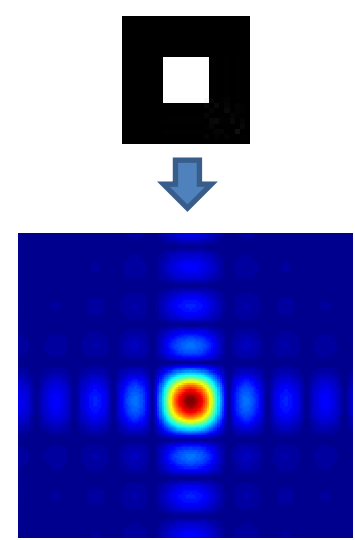
Perception

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



Things to Remember

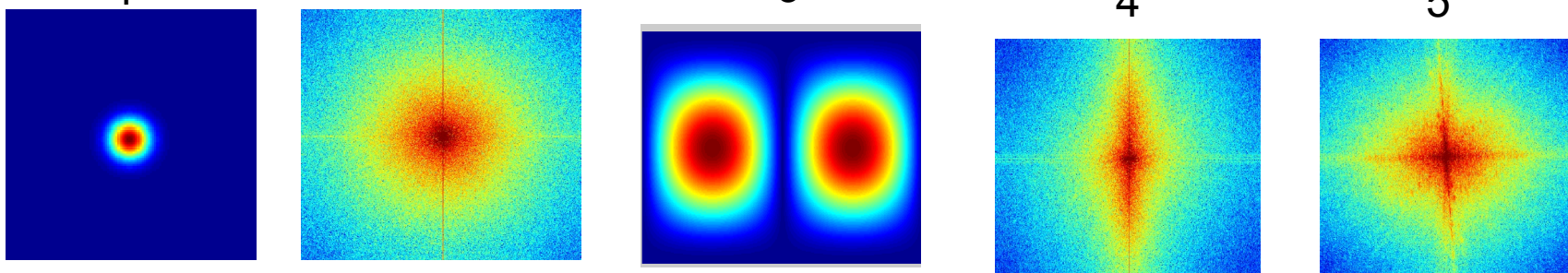
- Sometimes it makes sense to think of images and filtering in the frequency domain
 - Fourier analysis
- Can be faster to filter using FFT for large images ($N \log N$ vs. N^2 for auto-correlation)
- Images are mostly smooth
 - Basis for compression
- Remember to low-pass before sampling



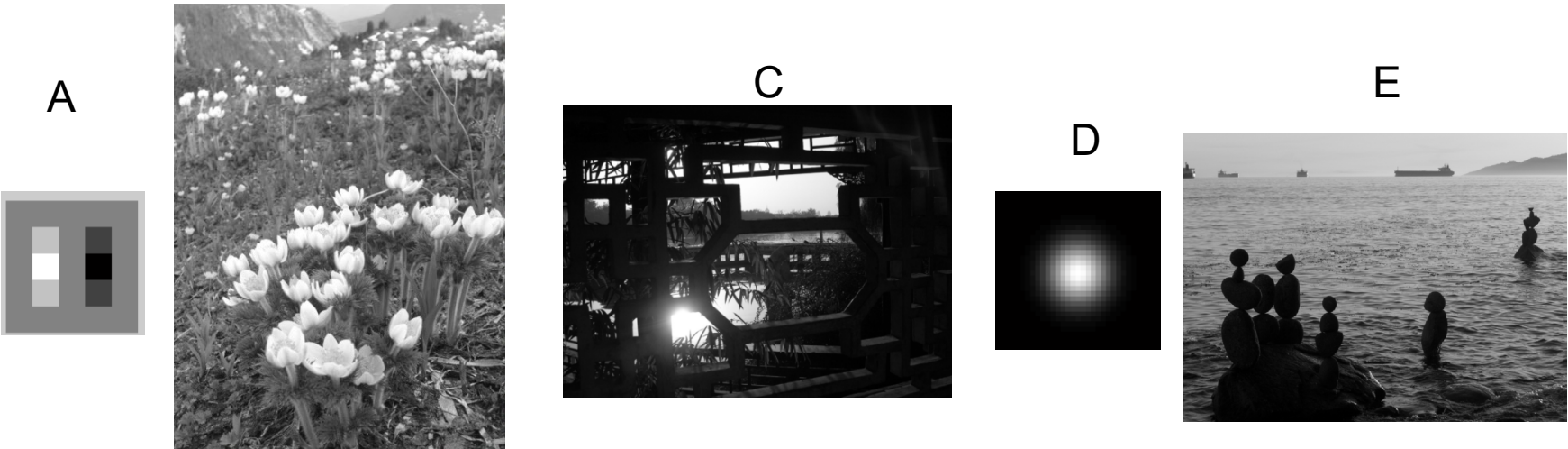
Take-home question

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

1 2 3 4 5



A B C D E



The task is to match the spatial domain images (A-E) to the Fourier magnitude images (1-5). The Fourier magnitude images show the frequency content of the spatial domain images. Image 1 is a single central peak, corresponding to a constant image (A). Image 2 is a central peak with a vertical line, corresponding to a periodic pattern in the vertical direction (B). Image 3 is two side-by-side peaks, corresponding to a periodic pattern in the horizontal direction (C). Image 4 is a central peak with a vertical line and a horizontal line, corresponding to a periodic pattern in both directions (D). Image 5 is a central peak with both vertical and horizontal lines, corresponding to a periodic pattern in both directions (E).

Next class: applications of filtering

- Denoising
- Template matching
- Image pyramids
- Compression