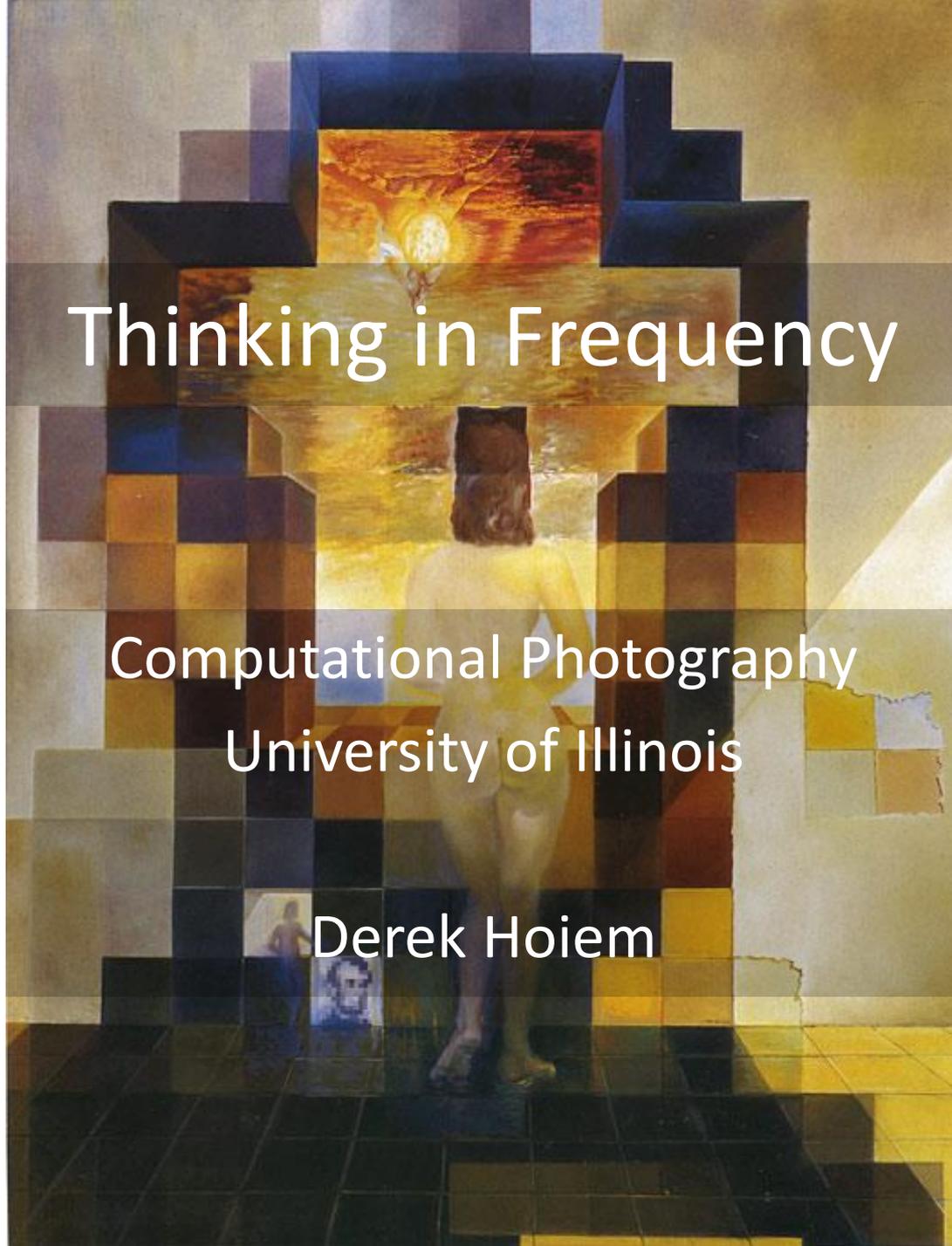


Thinking in Frequency

Computational Photography
University of Illinois

Derek Hoiem



Reminder

- Matlab/algebra tutorial today, 5pm, SC3403

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 negative value otherwise
2. Write down a filter that will compute the gradient in the x-direction:

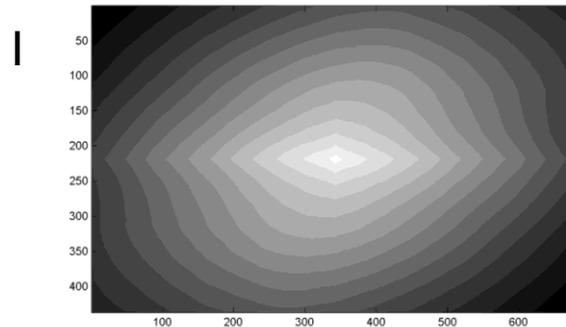
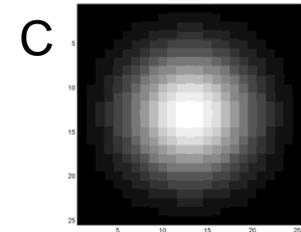
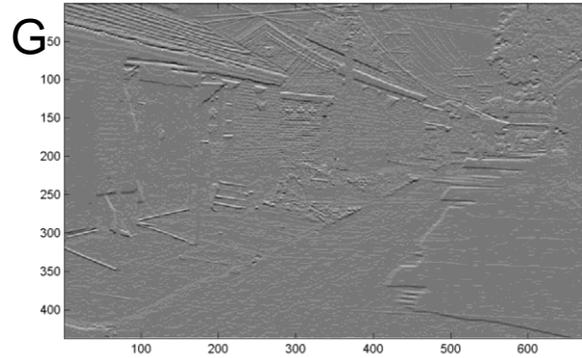
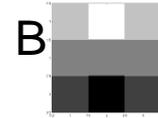
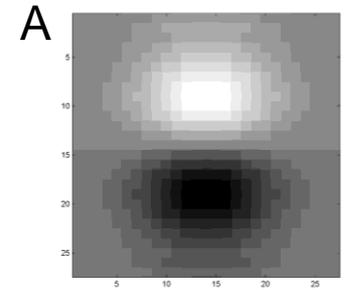
$$\text{grad}_x(y, x) = \text{im}(y, x+1) - \text{im}(y, x) \quad \text{for each } x, y$$

Review: questions

3. Fill in the blanks:

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

Filtering Operator \swarrow

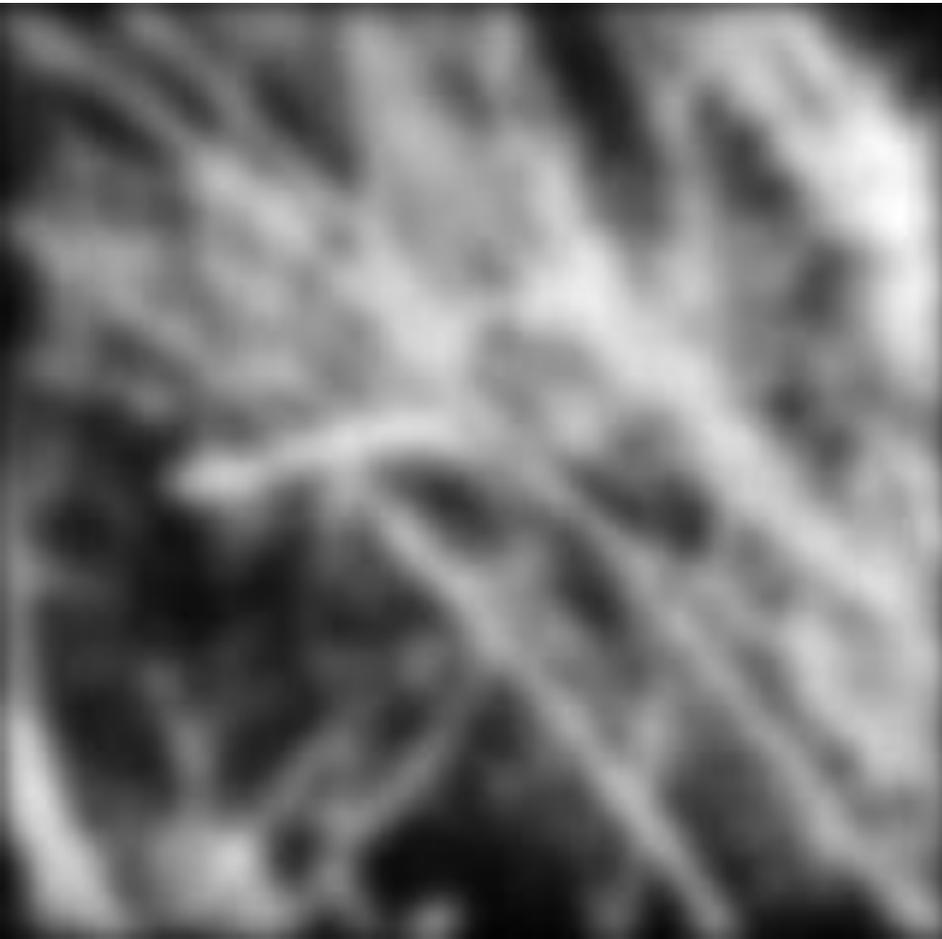


Today's Class

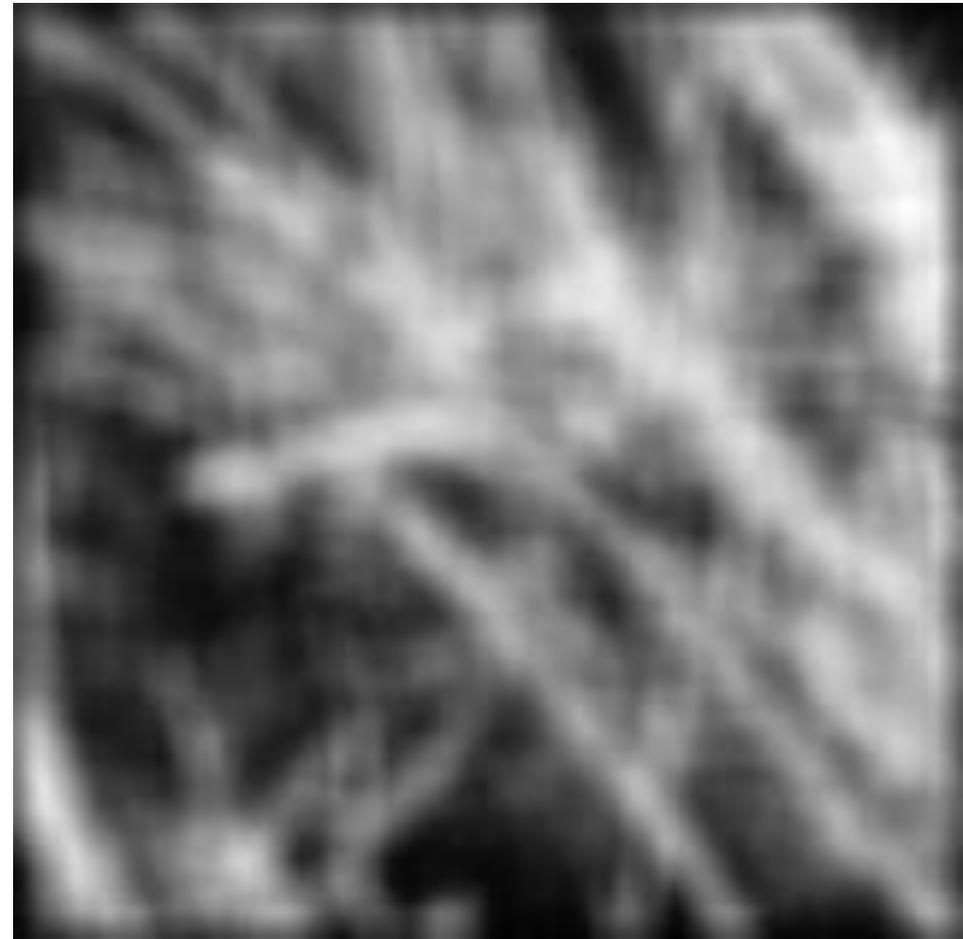
- Fourier transform and frequency domain
 - Frequency view of filtering
 - Another look at hybrid images
 - Sampling

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

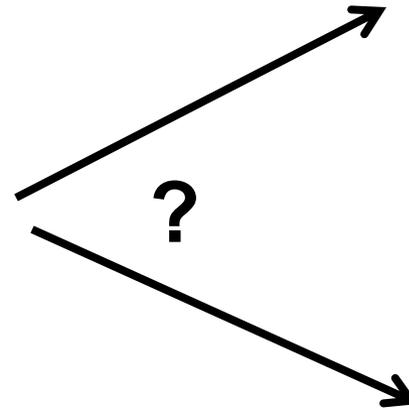
Gaussian



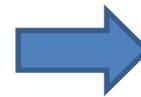
Box filter



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

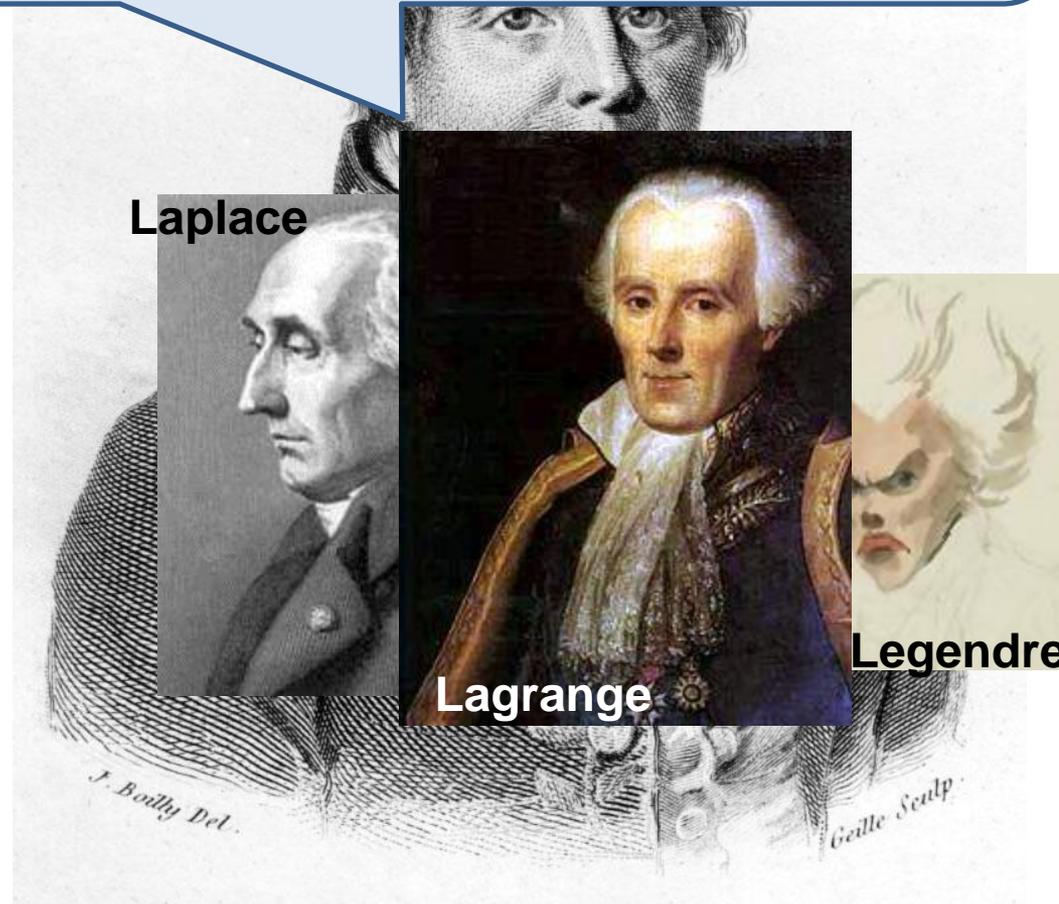
Jean Baptiste Joseph Fourier (1768-1830)

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

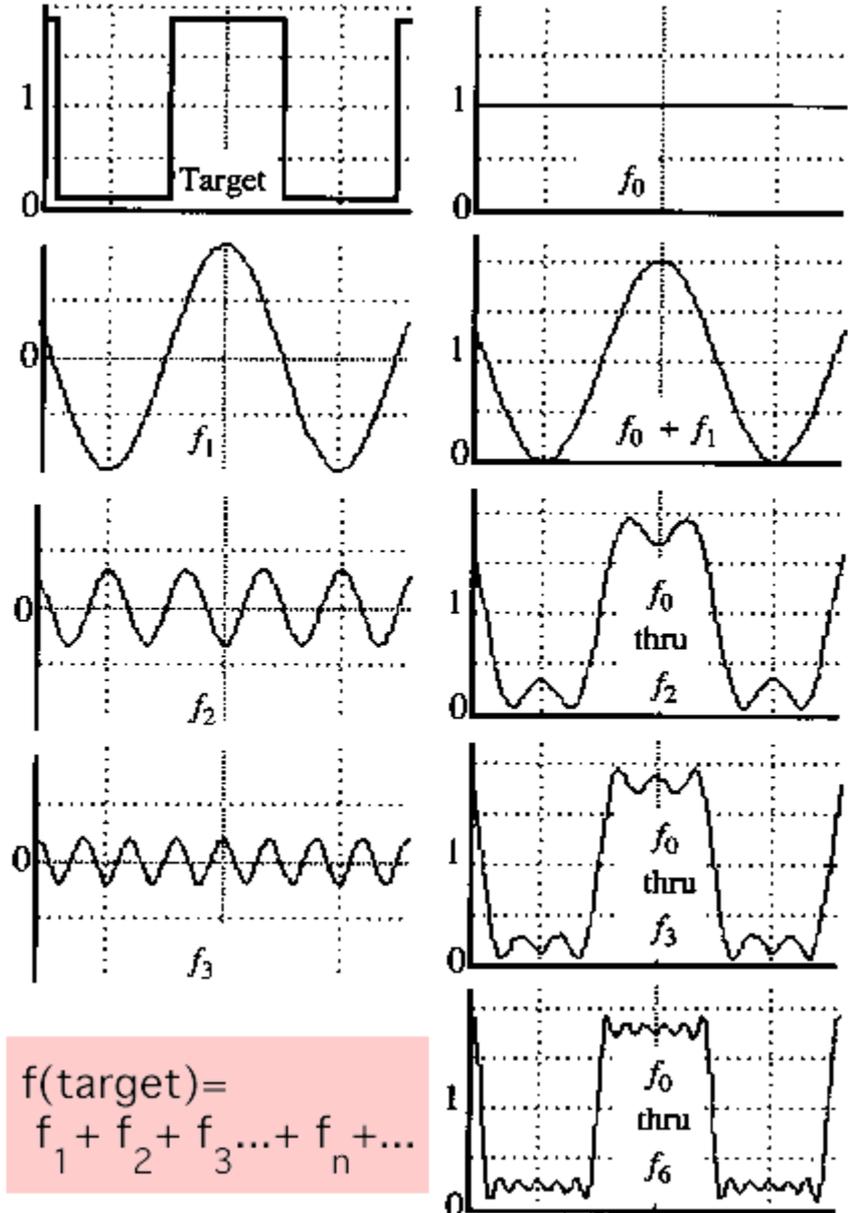


A sum of sines

Our building block:

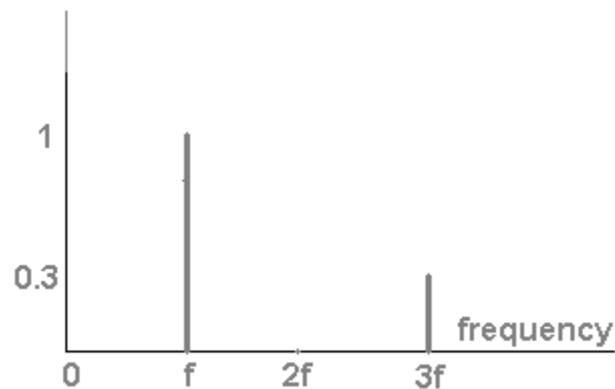
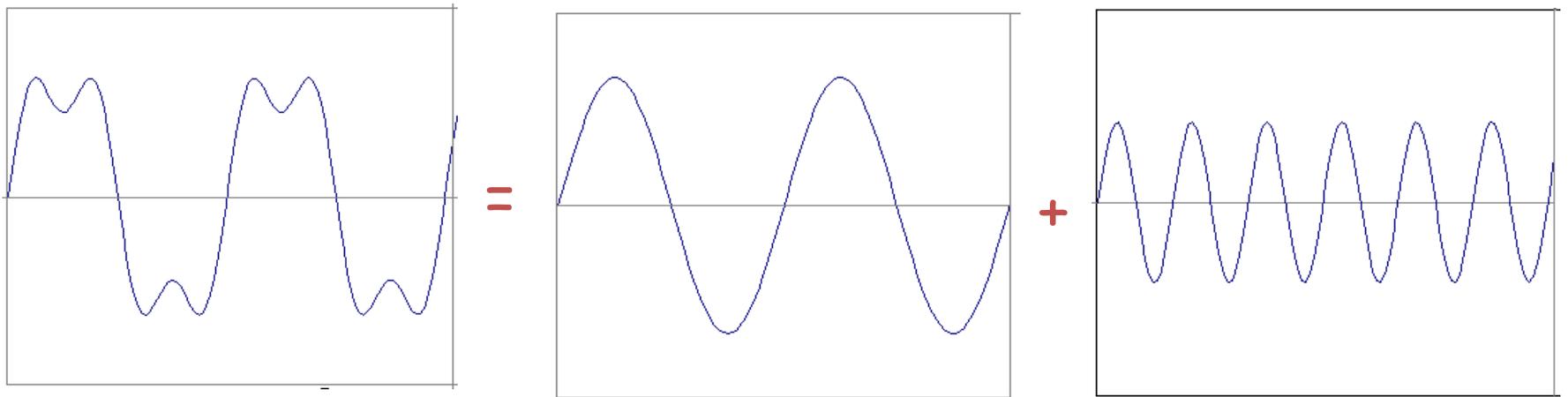
$$A \sin(\omega x + \phi)$$

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

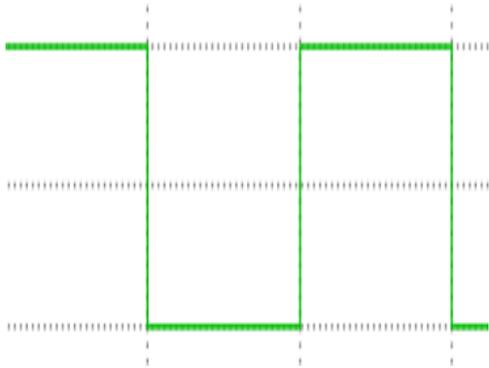


Frequency Spectra

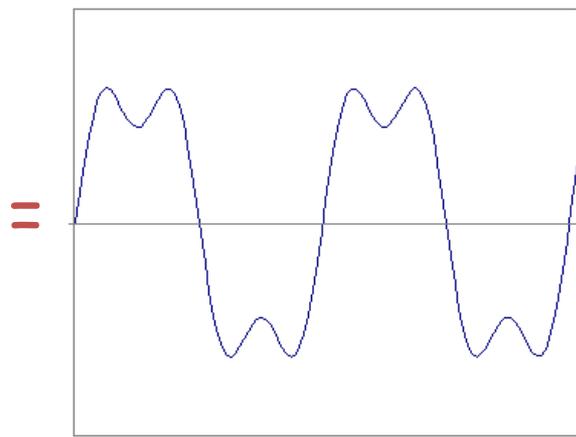
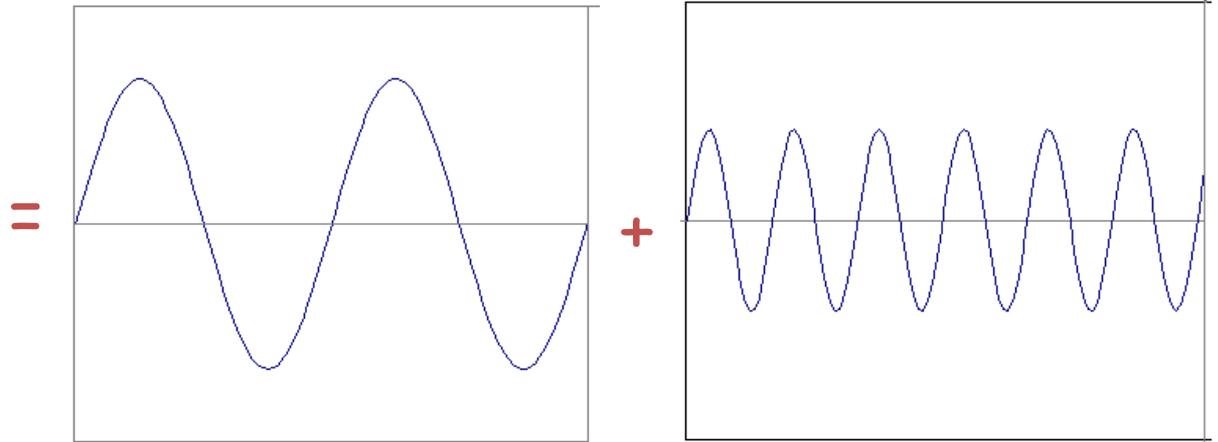
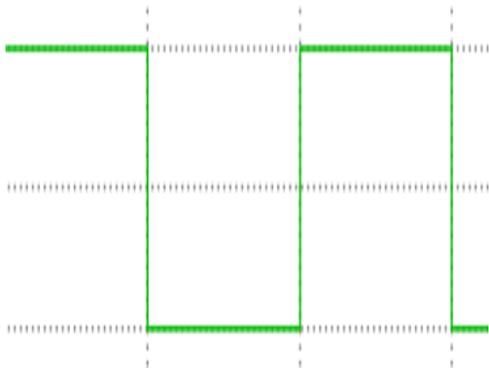
- example : $g(t) = \sin(2\pi f t) + (1/3)\sin(2\pi(3f) t)$



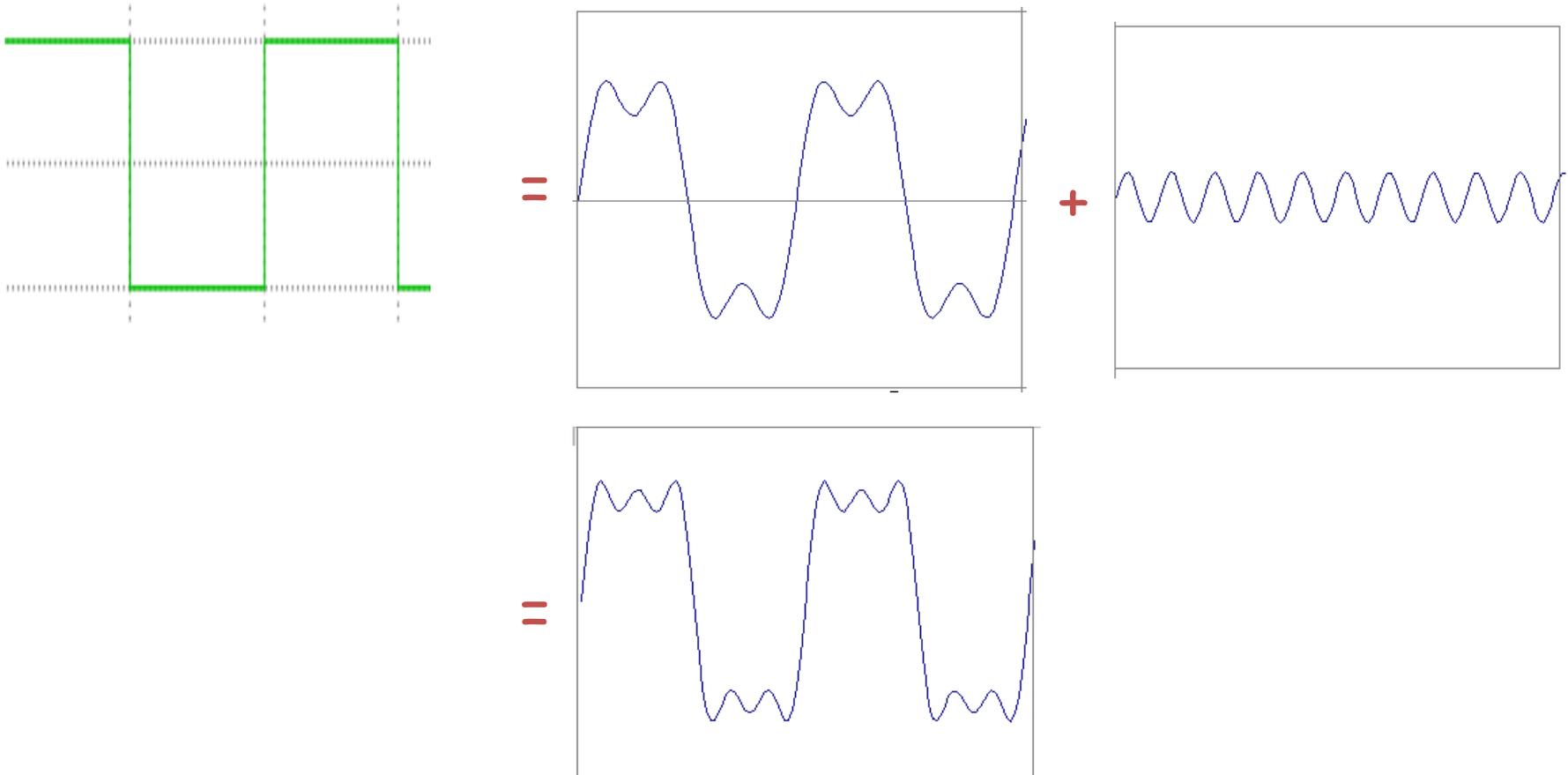
Frequency Spectra



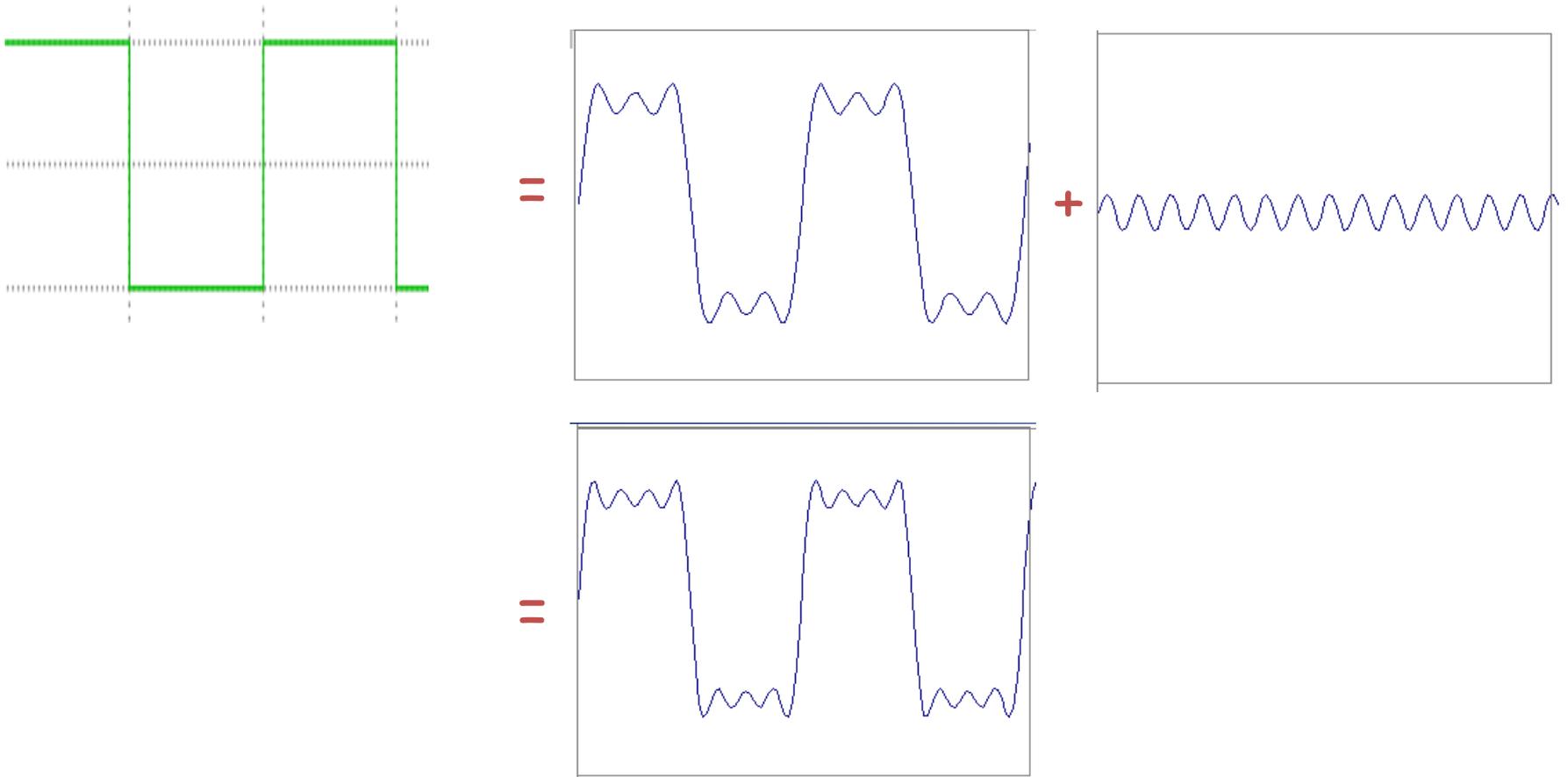
Frequency Spectra



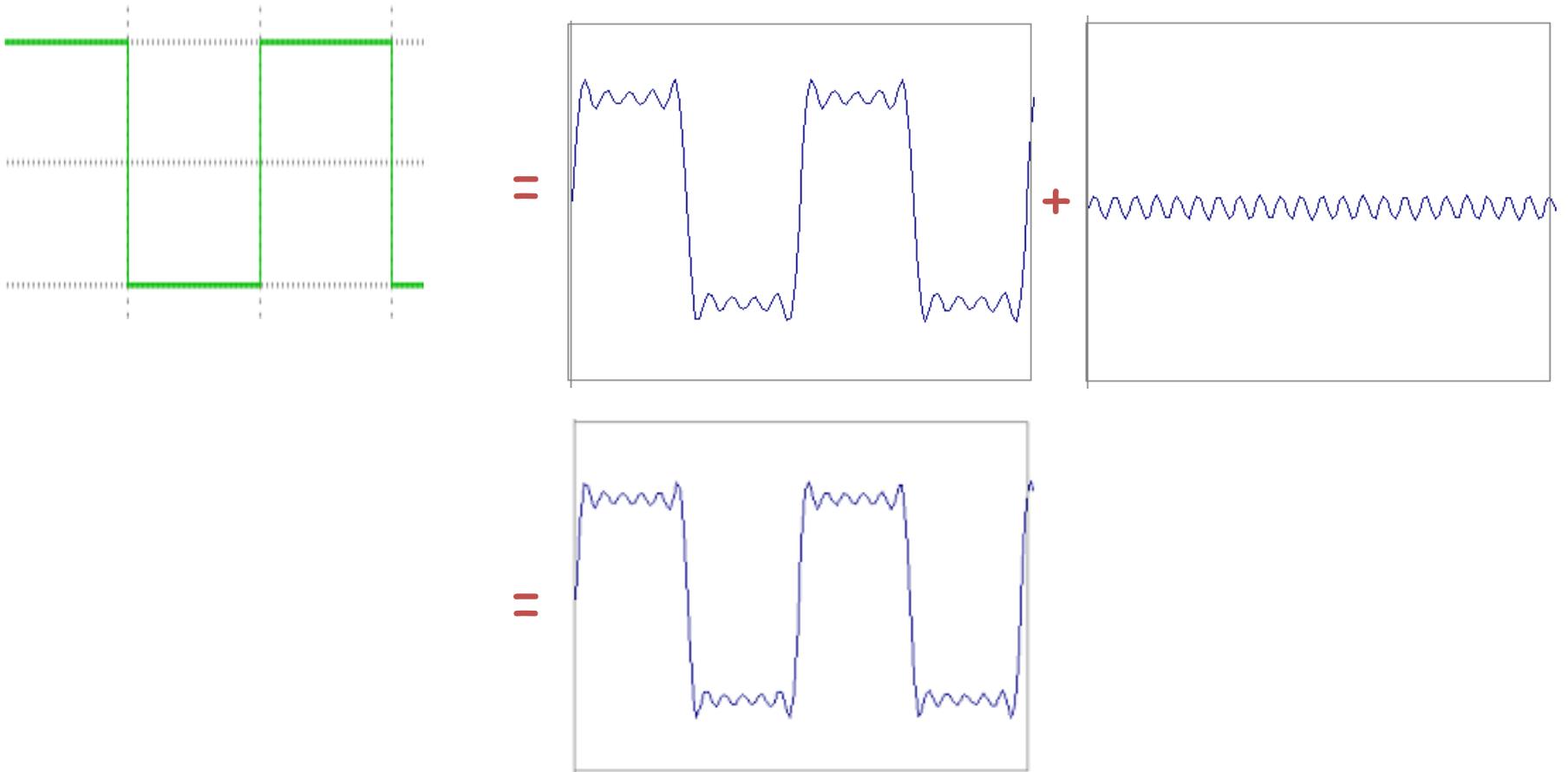
Frequency Spectra



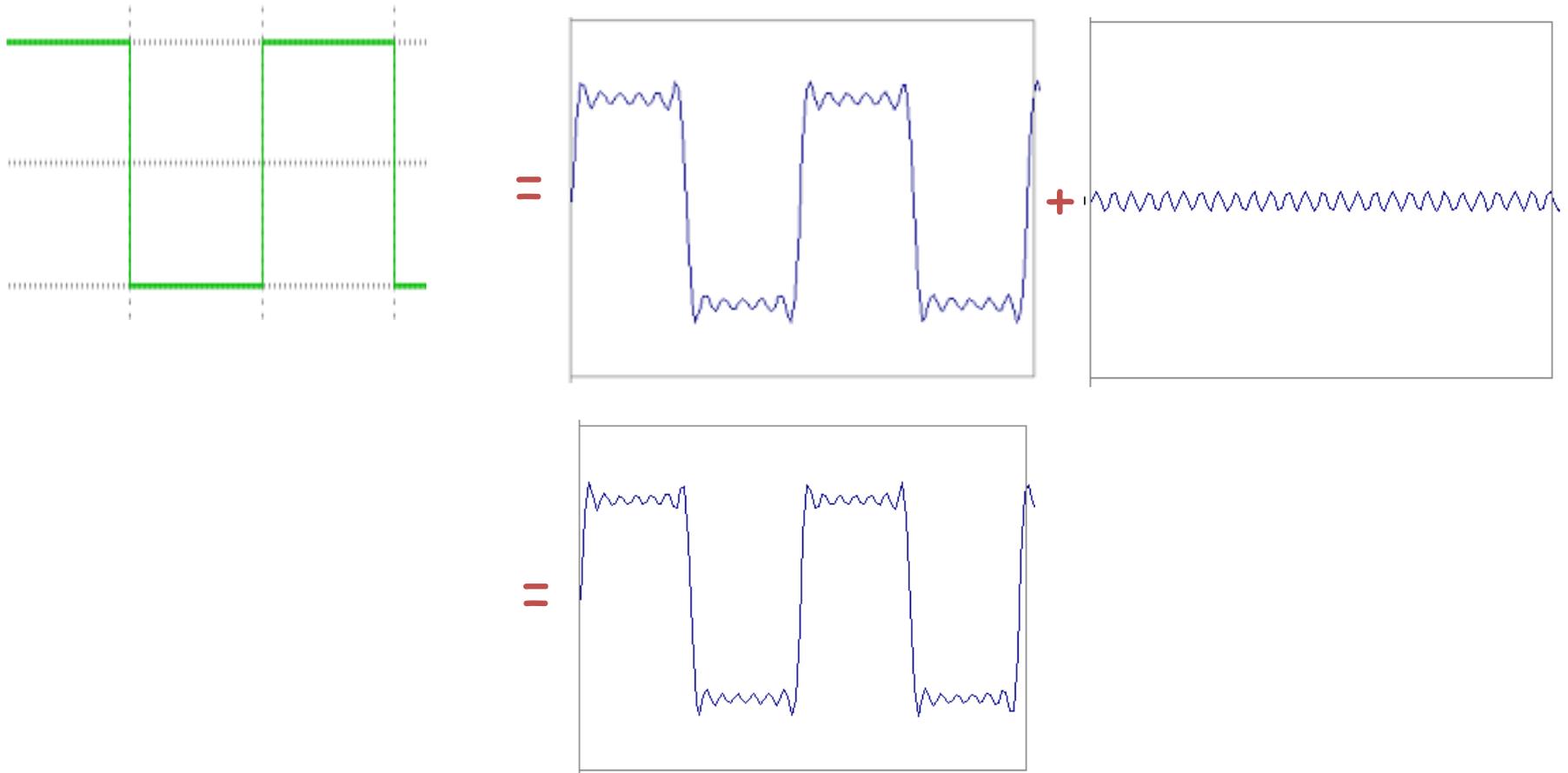
Frequency Spectra



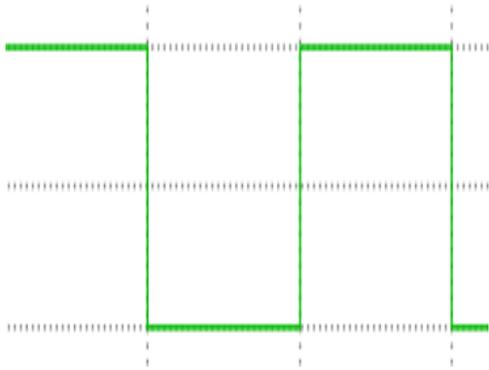
Frequency Spectra



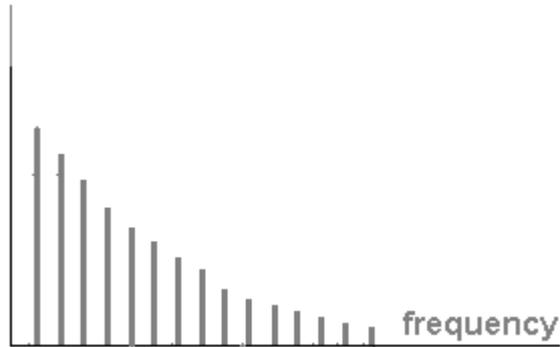
Frequency Spectra



Frequency Spectra

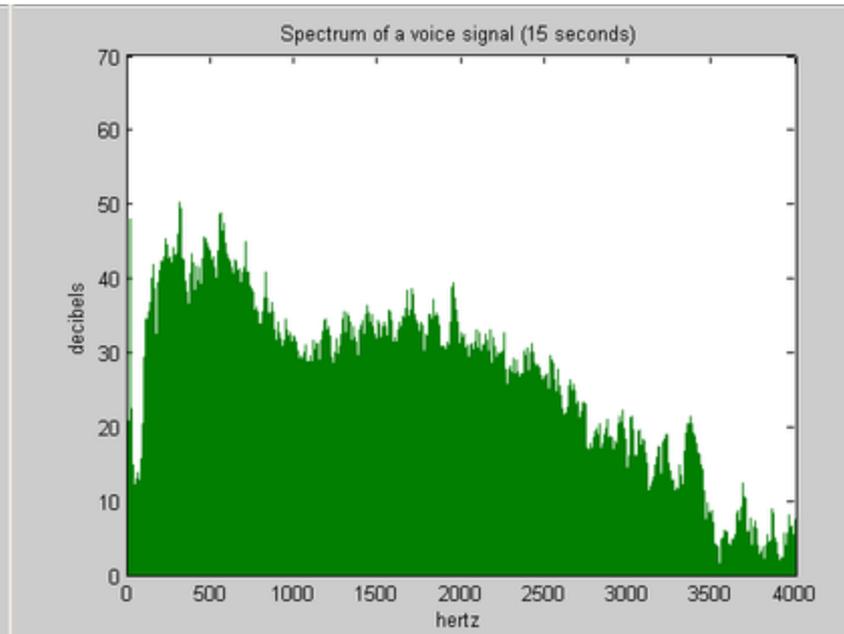
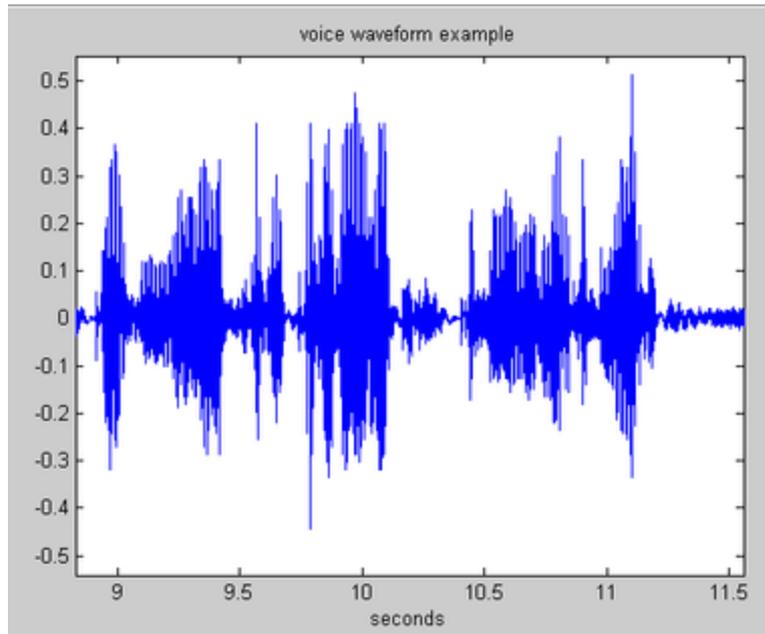


$$= A \sum_{k=1}^{\infty} \frac{1}{k} \sin(2\pi kt)$$



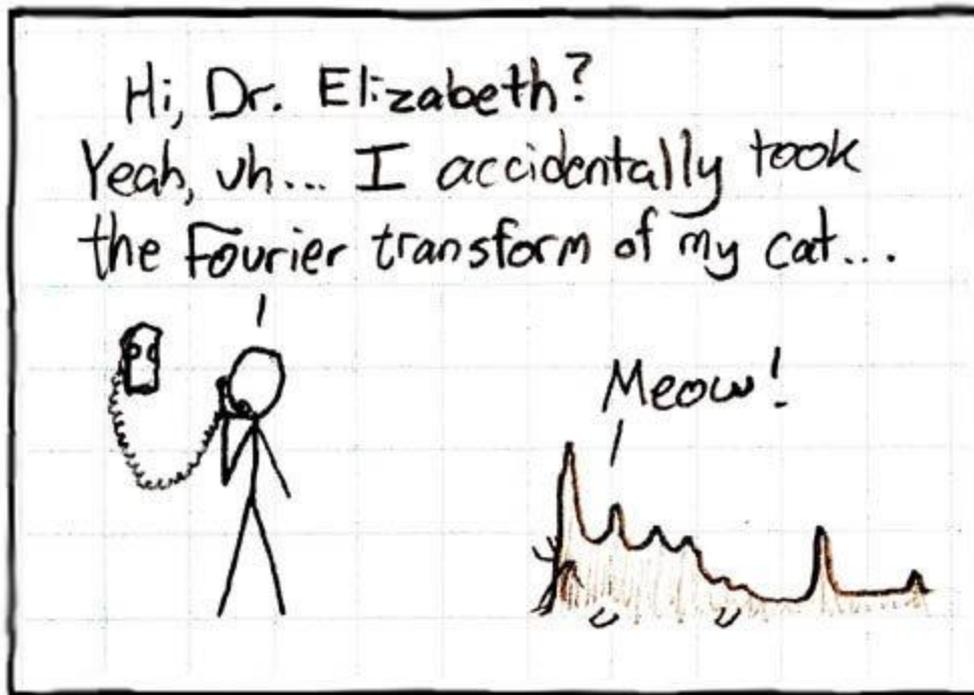
Example: Music

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



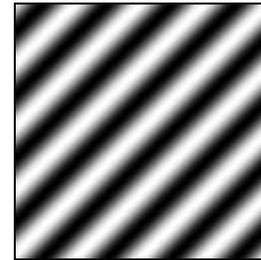
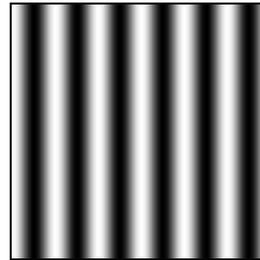
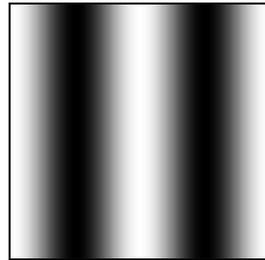
Other signals

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

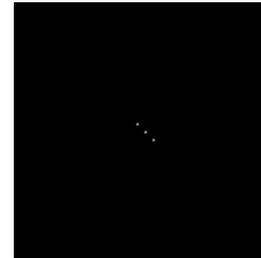
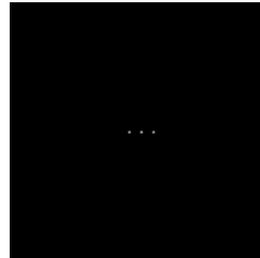
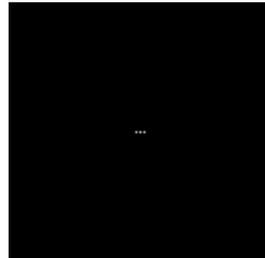


Fourier analysis in images

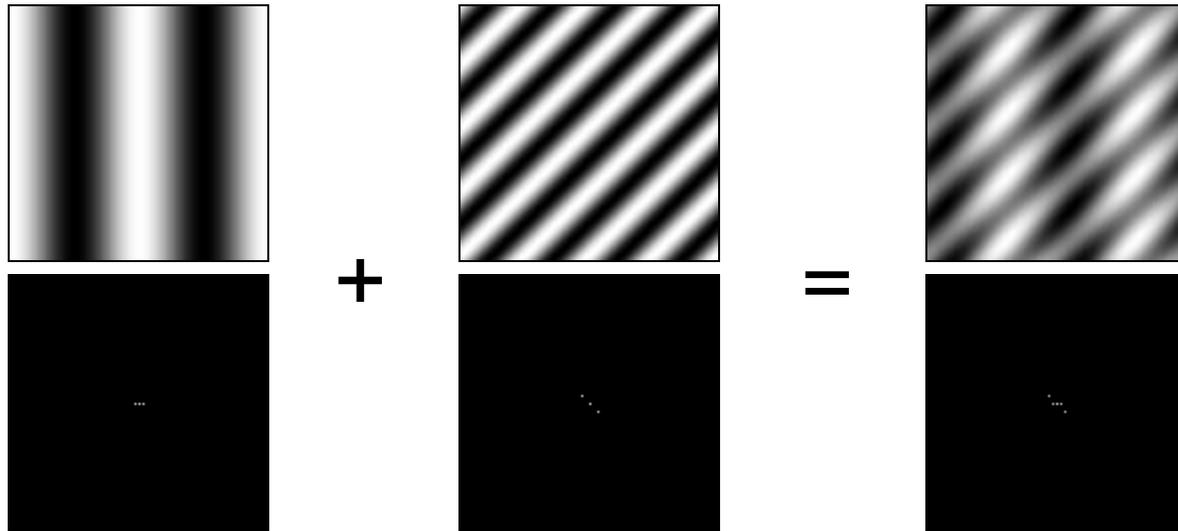
Intensity Image



Fourier Image



Signals can be composed



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 real and 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

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

[Fast Fourier Transform](#) (FFT): $N \log N$

The Convolution Theorem

- The Fourier transform of the convolution of two functions is the 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!

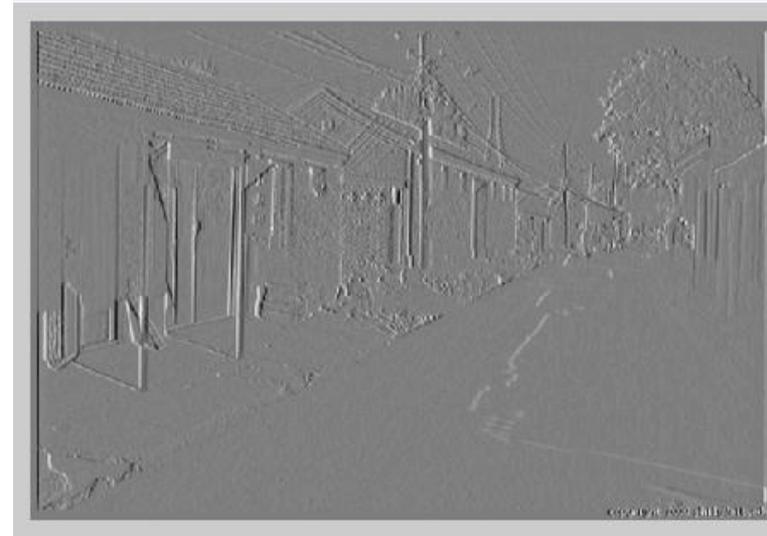
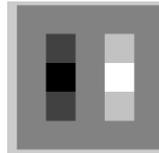
Properties of Fourier Transforms

- 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

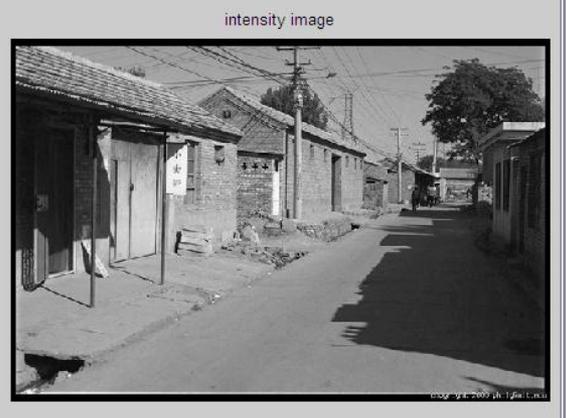
Filtering in spatial domain

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

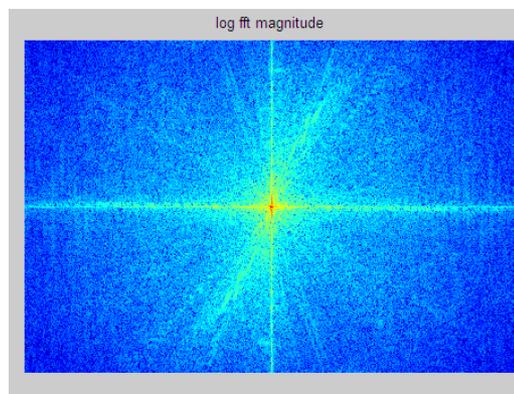
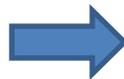
intensity image



Filtering in frequency domain

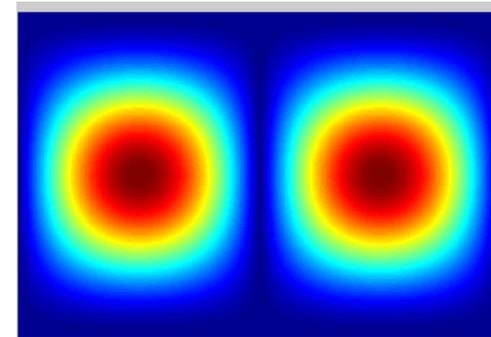


FFT

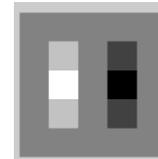


\times

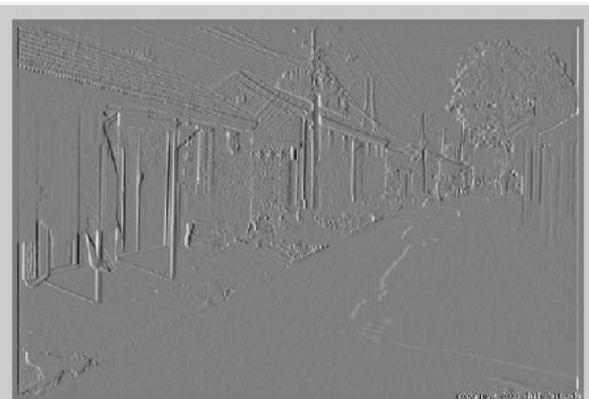
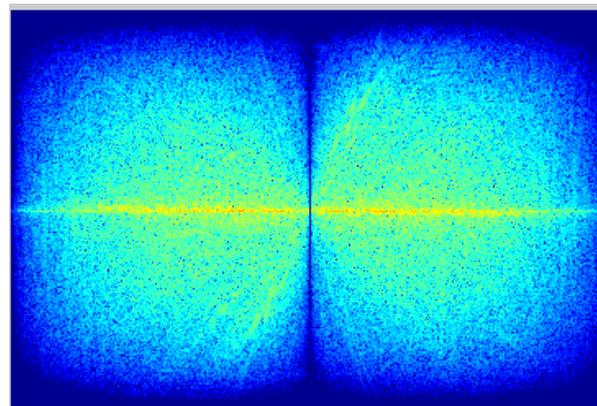
$=$



FFT



Inverse FFT



Fourier Matlab demo

FFT in Matlab

- Filtering with fft

```
im = ... % "im" should be a gray-scale floating point image
[imh, imw] = size(im);
fftsize = 1024; % should be order of 2 (for speed) and include padding
im_fft = fft2(im, fftsize, fftsize); % 1) fft im with padding
hs = 50; % filter half-size
fil = fspecial('gaussian', hs*2+1, 10);
fil_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 = ifft2(im_fil_fft); % 4) inverse fft2
im_fil = im_fil(1+hs:size(im,1)+hs, 1+hs:size(im, 2)+hs); % 5) remove padding
```

- Displaying with fft

```
figure(1), imagesc(log(abs(fftshift(im_fft)))), axis image, colormap jet
```

Questions

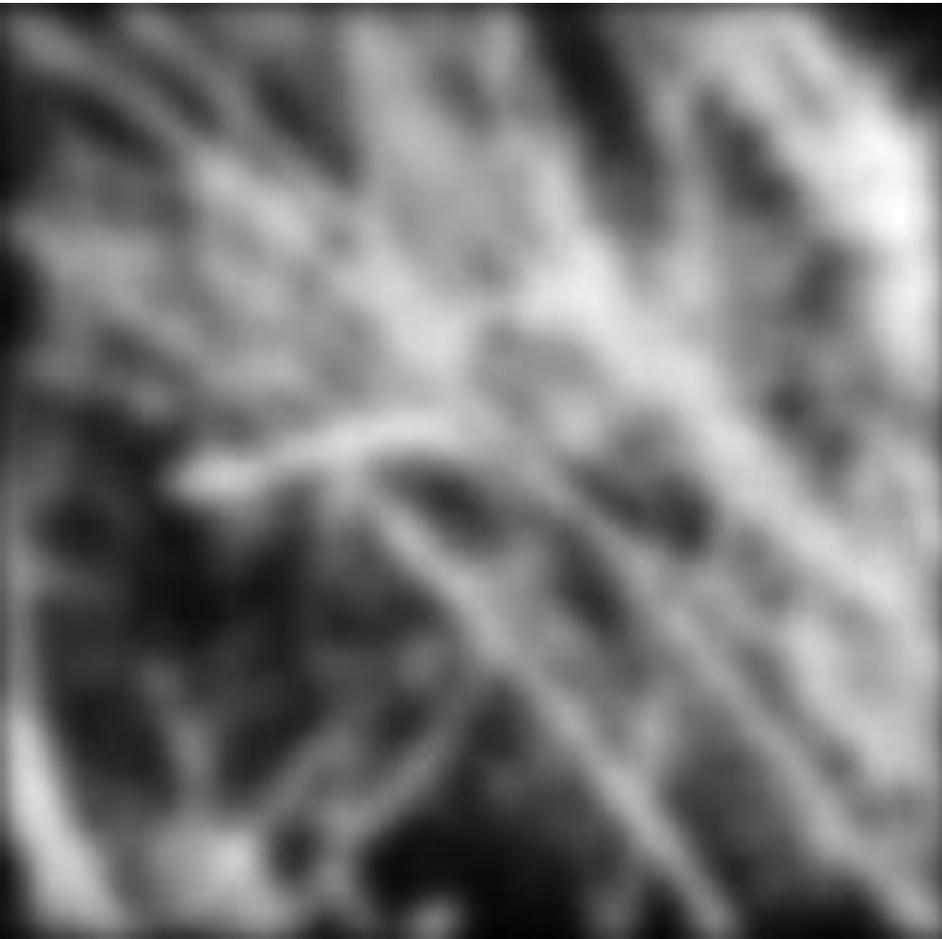
Which has more information, the phase or the magnitude?

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

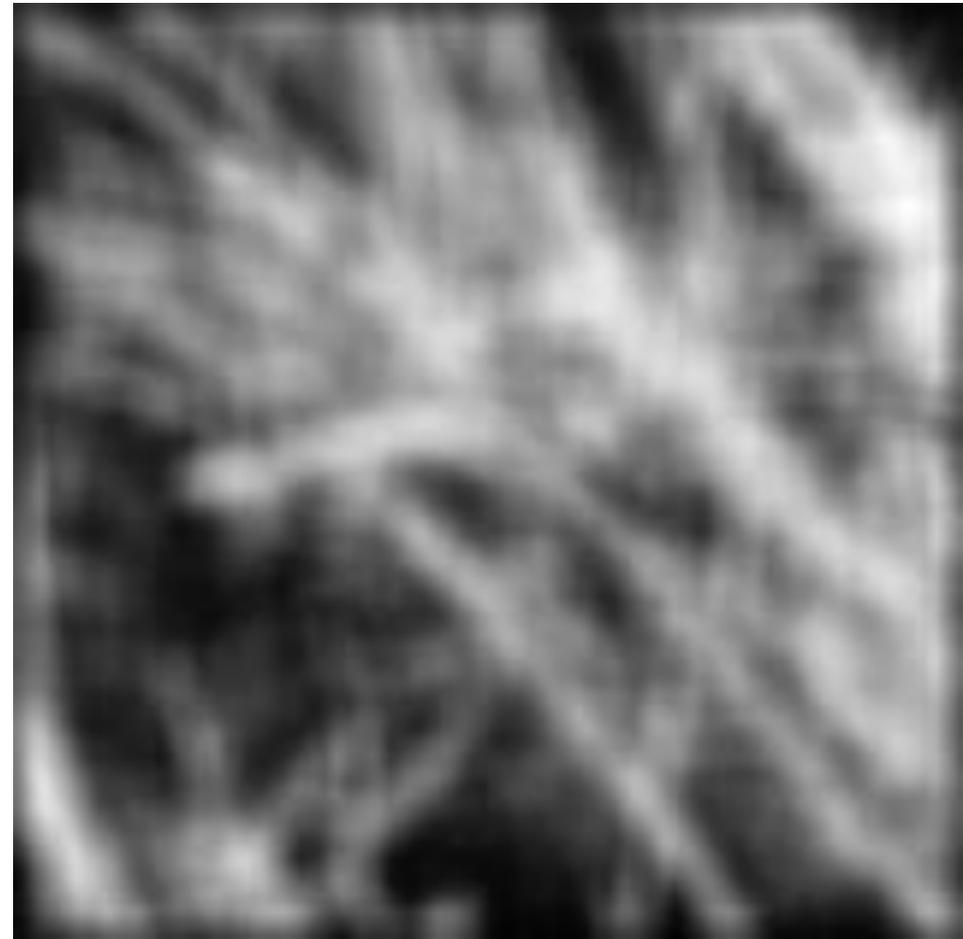
Filtering

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

Gaussian



Box filter

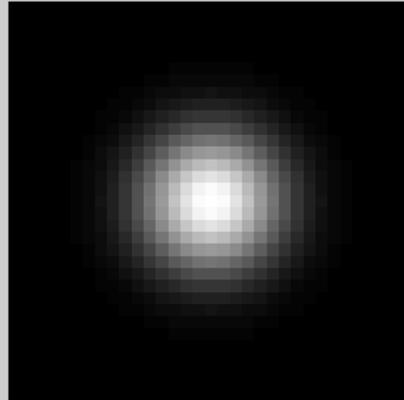


Gaussian

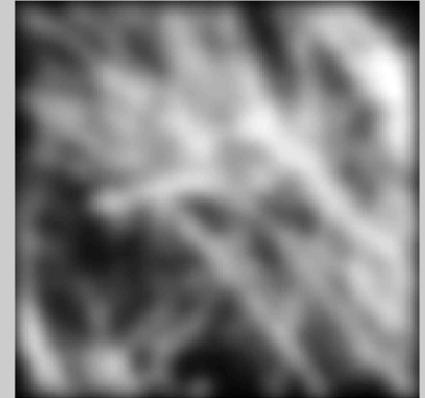
intensity image



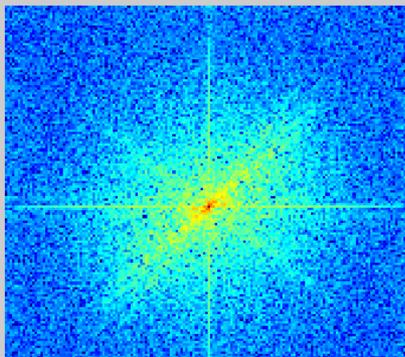
filter: gaussian



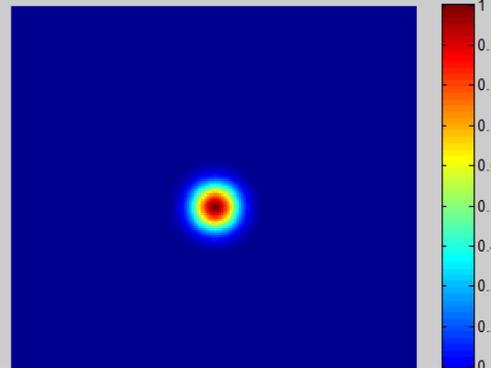
filtered image



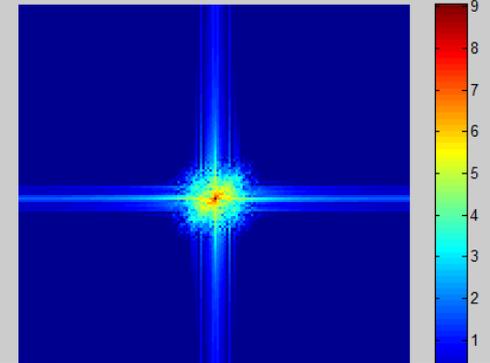
log fit magnitude of image



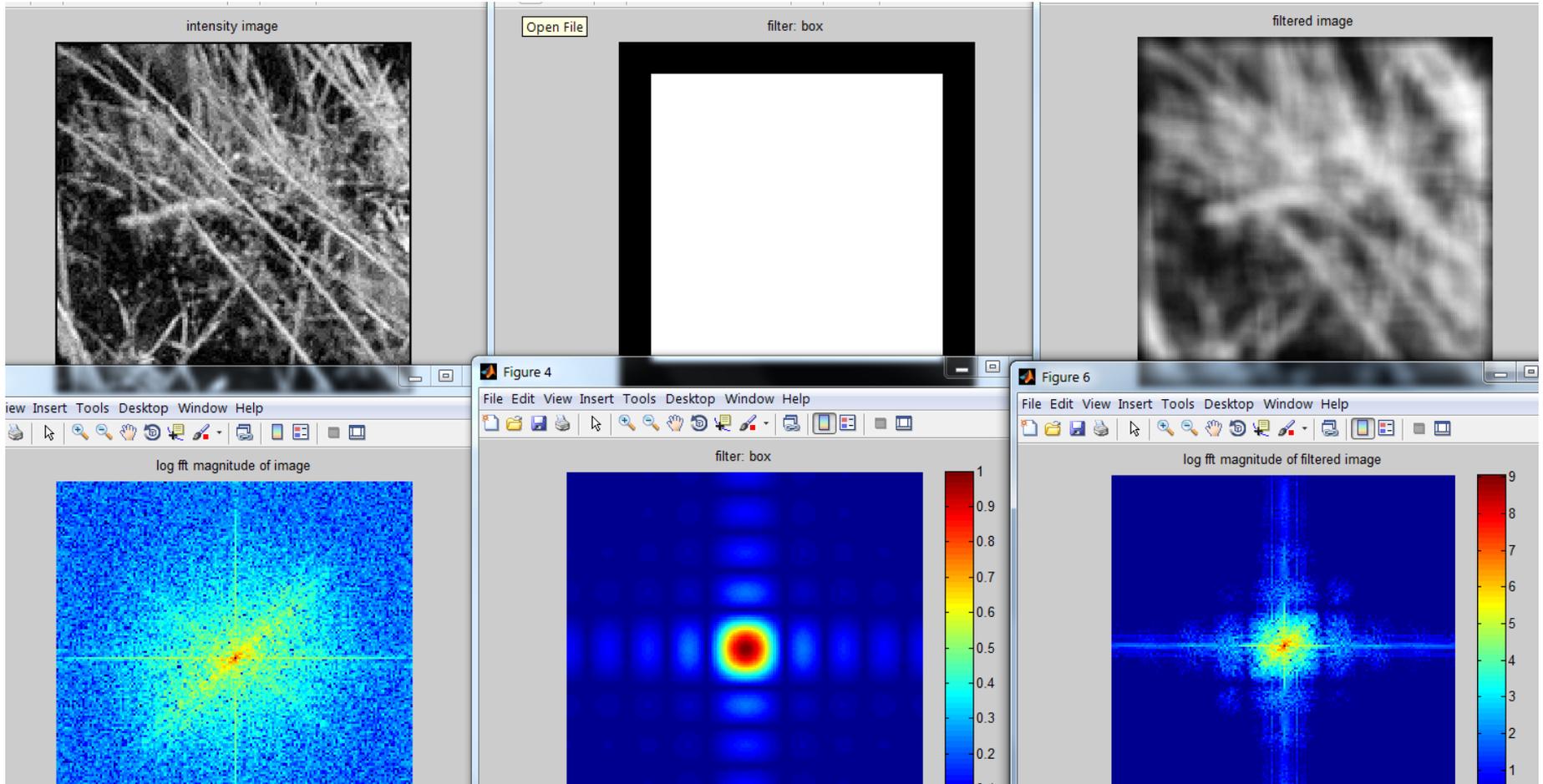
filter: gaussian



log fit magnitude of filtered image

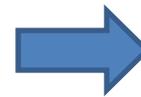


Box Filter



Sampling

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



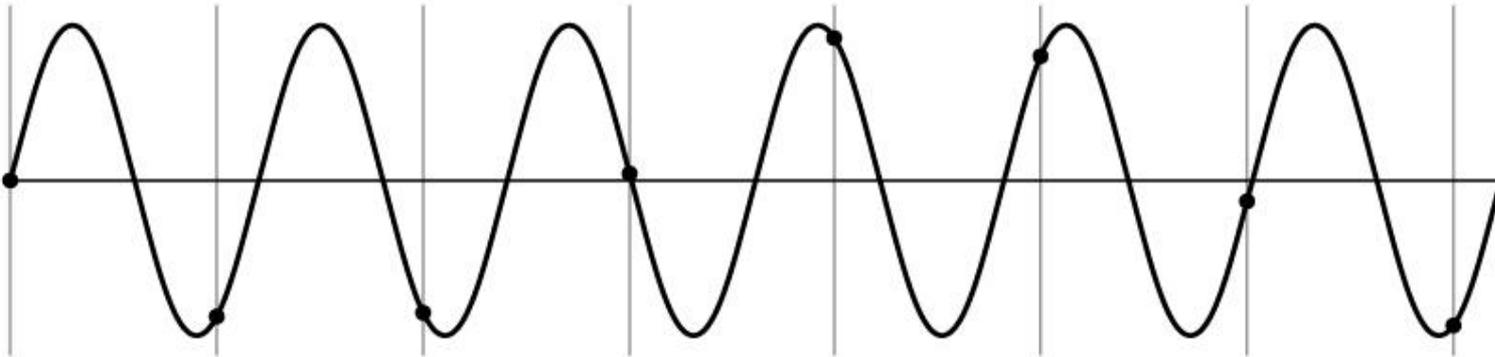
Subsampling by a factor of 2



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

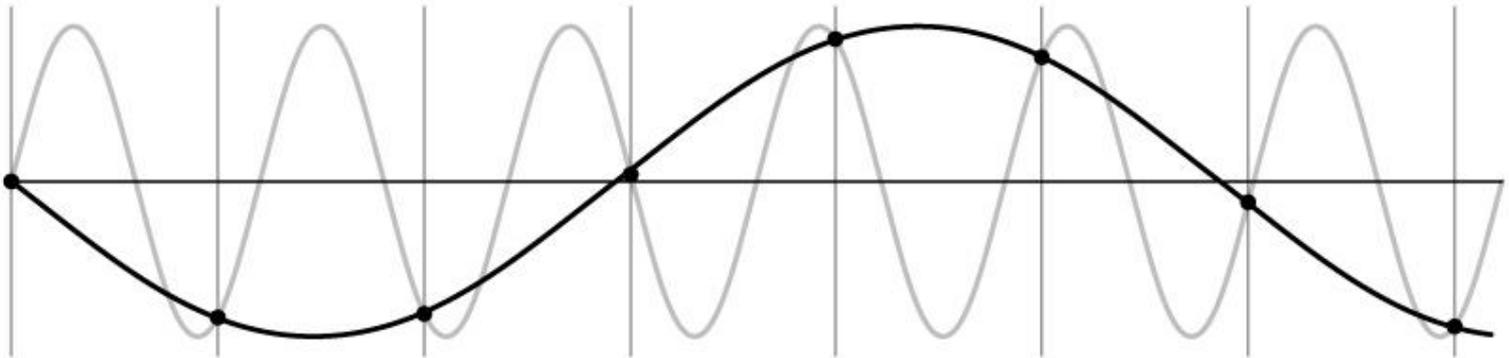
Aliasing problem

- 1D example (sinewave):



Aliasing problem

- 1D example (sinewave):



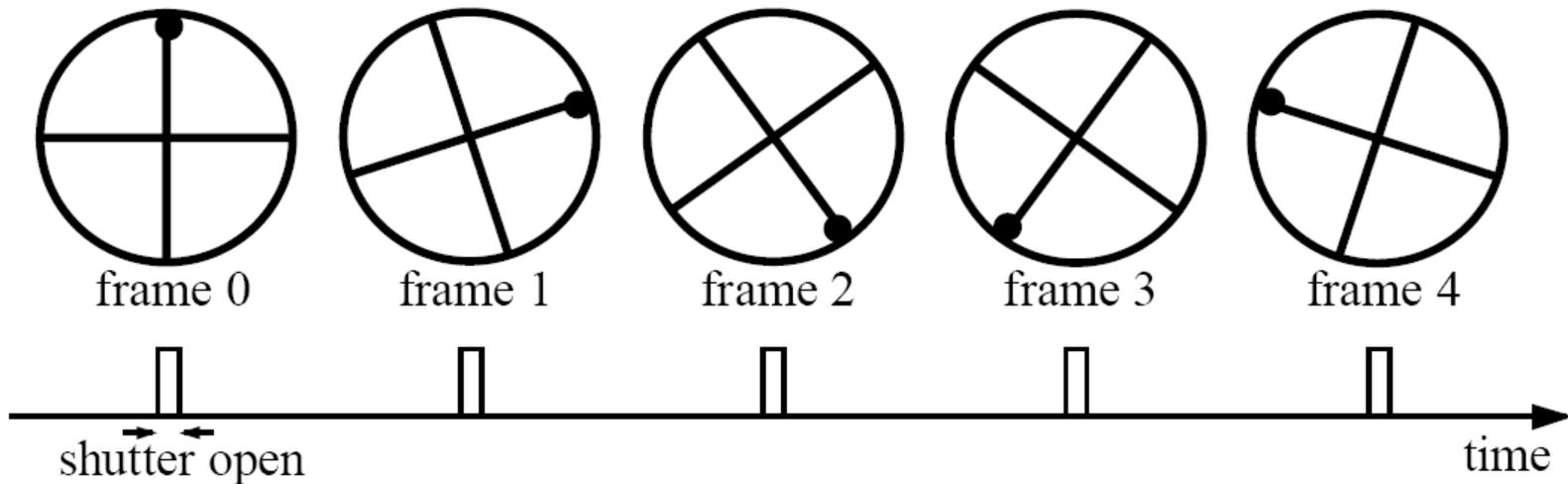
Aliasing problem

- 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

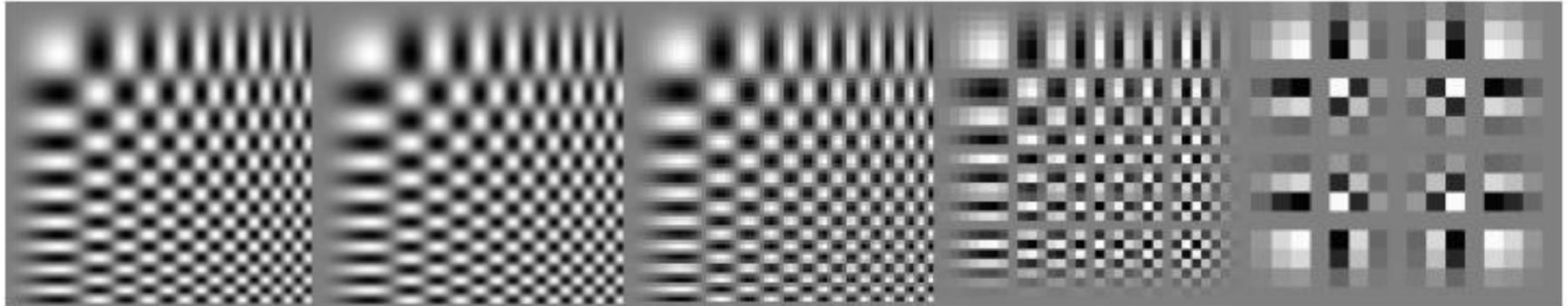
256x256

128x128

64x64

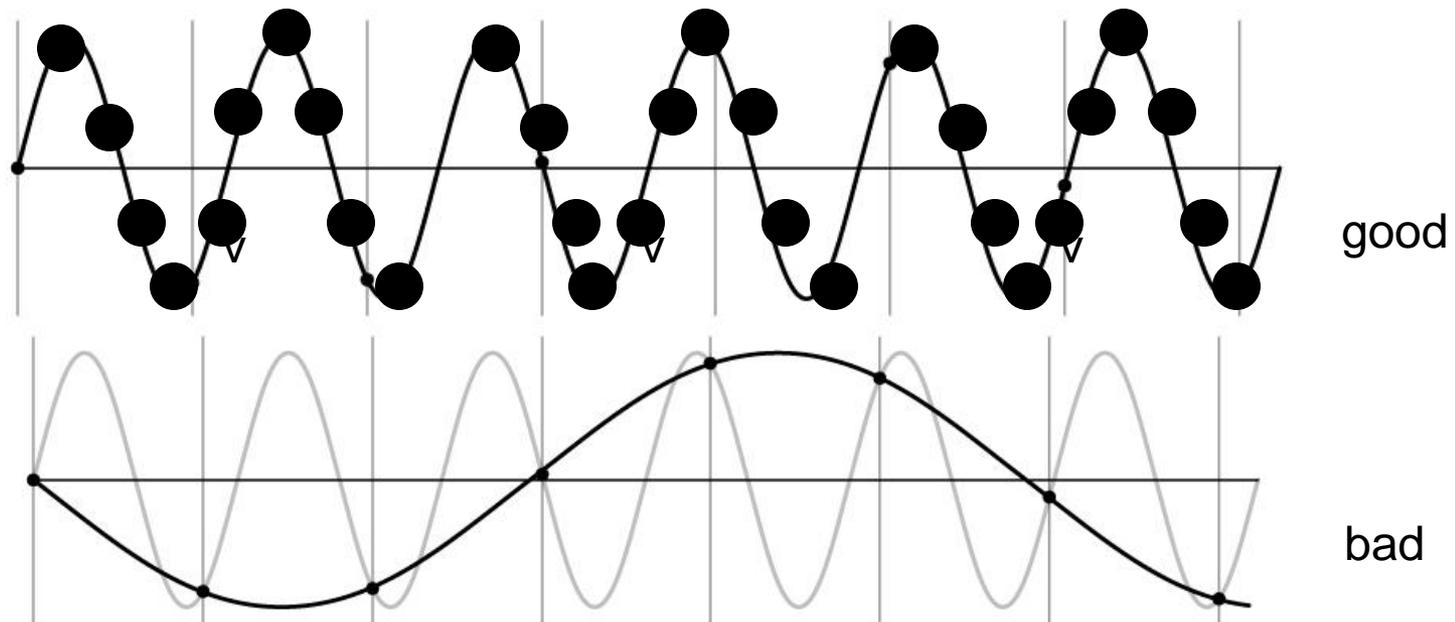
32x32

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

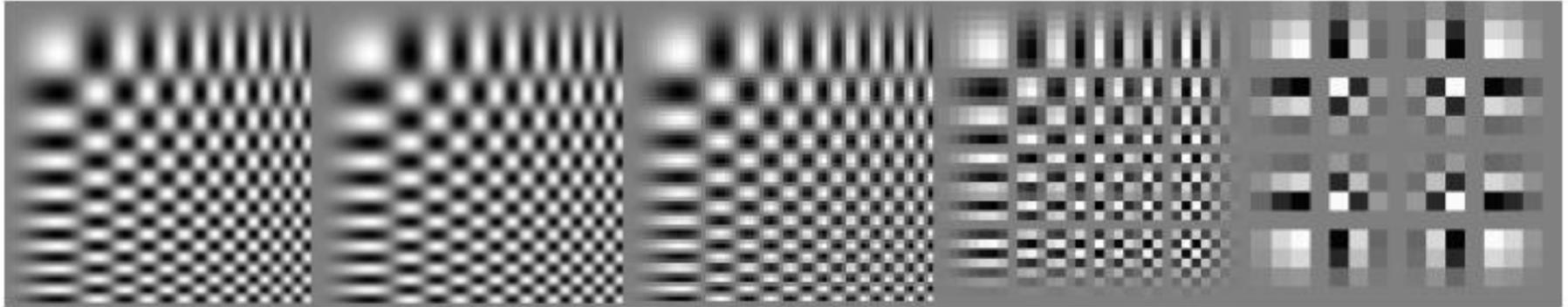
256x256

128x128

64x64

32x32

16x16



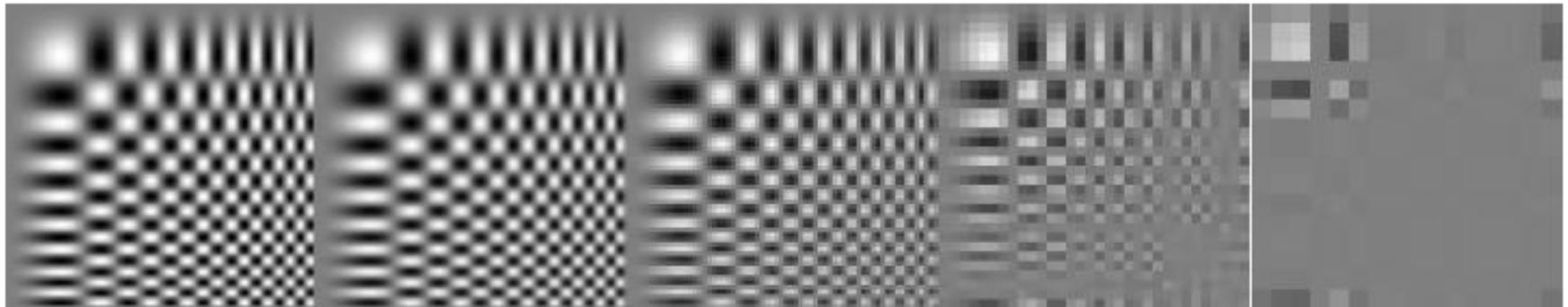
256x256

128x128

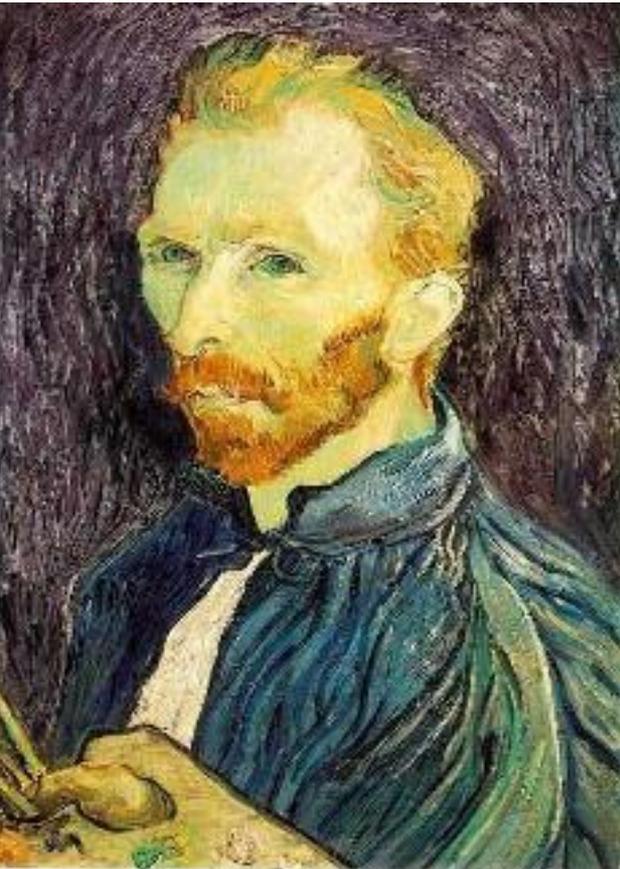
64x64

32x32

16x16



Subsampling without pre-filtering



1/2

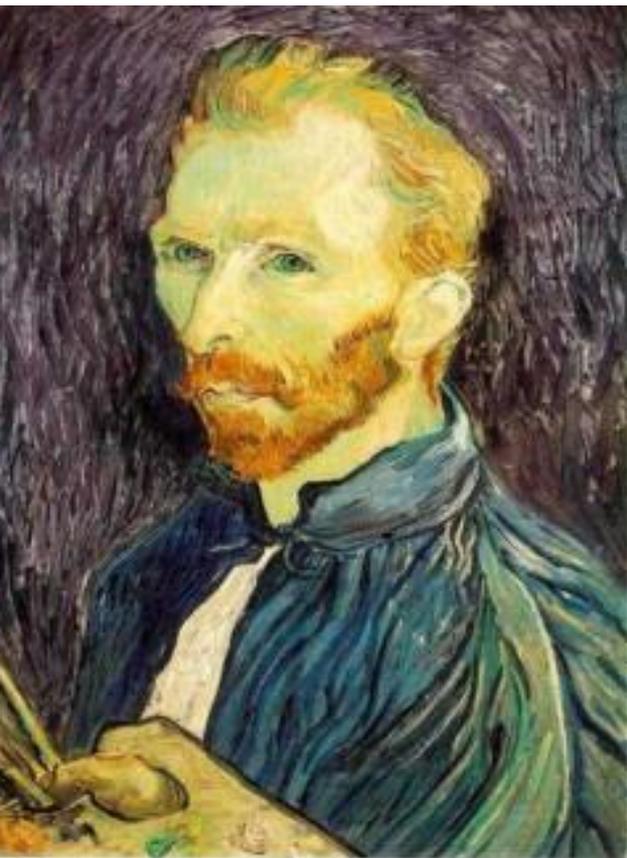


1/4 (2x zoom)

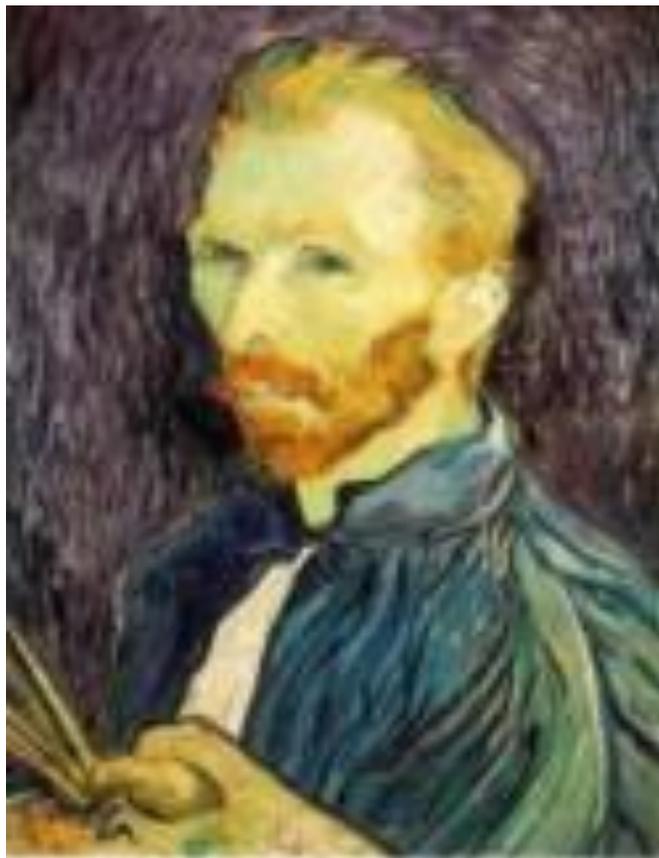


1/8 (4x zoom)

Subsampling with Gaussian pre-filtering



Gaussian $1/2$

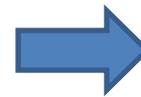


G $1/4$

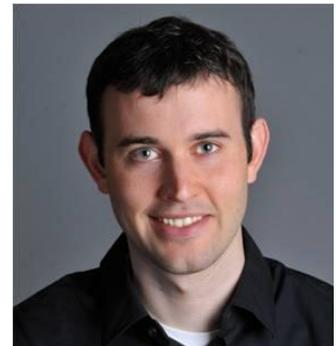
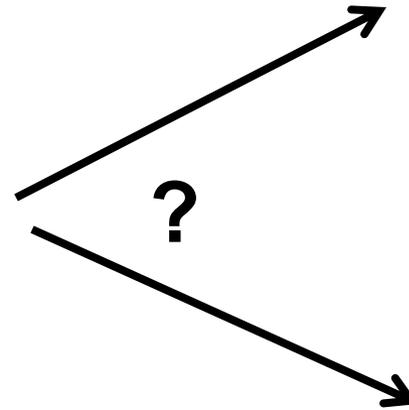


G $1/8$

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

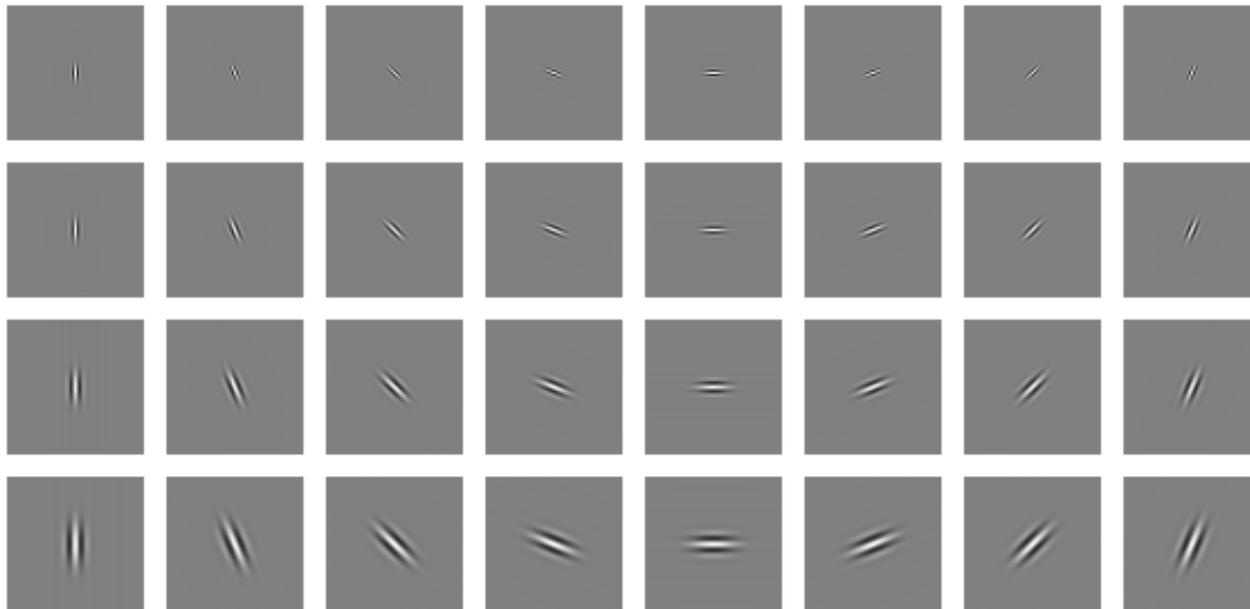


Why do we get different, distance-dependent interpretations of hybrid images?



Clues from Human Perception

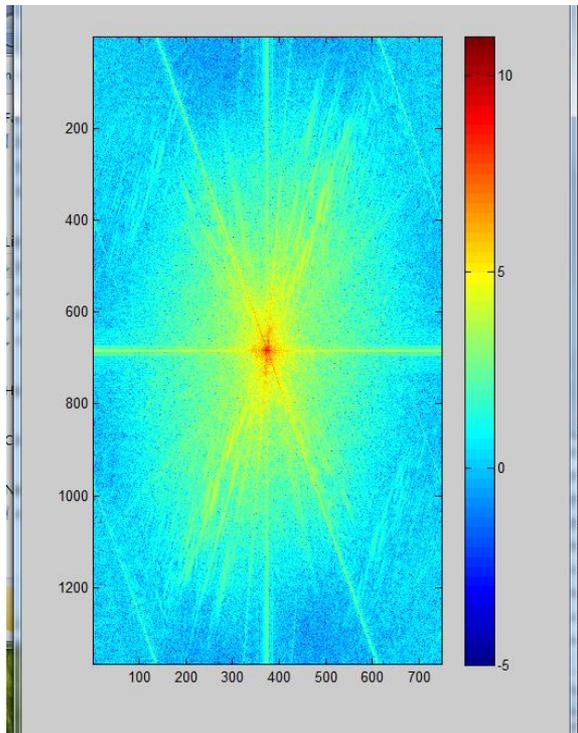
- Early processing in humans filters for various orientations and scales of frequency
- Perceptual cues in the mid frequencies dominate perception
- When we see an image from far away, we are effectively subsampling it



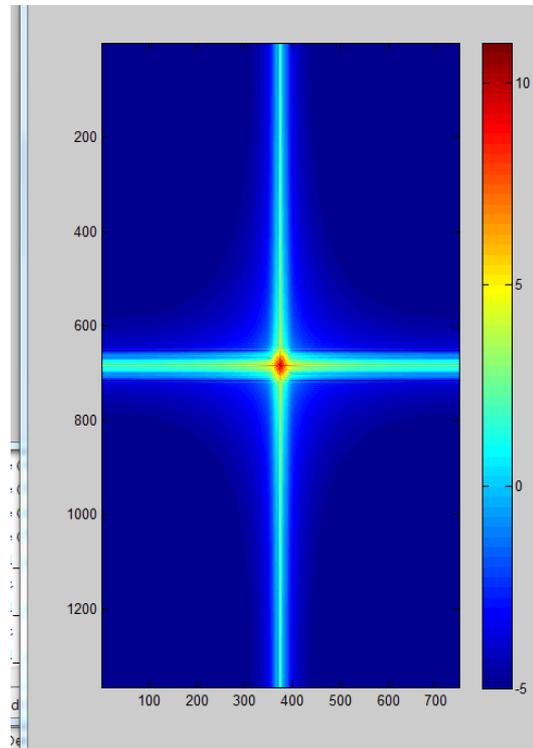
Early Visual Processing: Multi-scale edge and blob filters

Hybrid Image in FFT

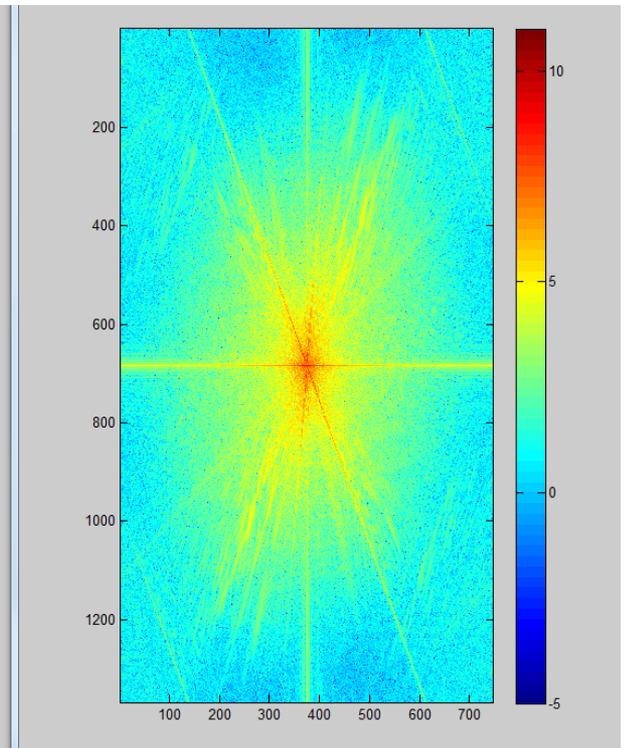
Hybrid Image



Low-passed Image

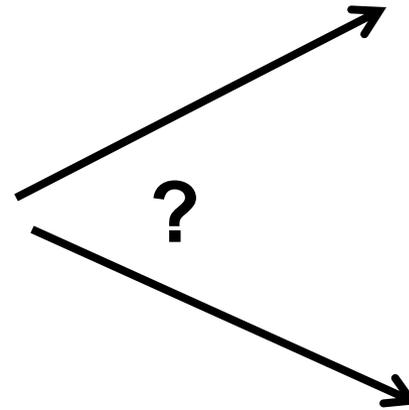


High-passed Image



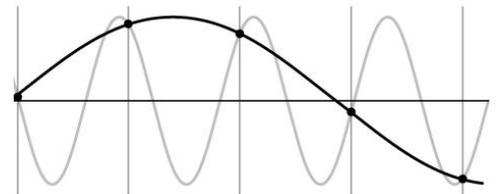
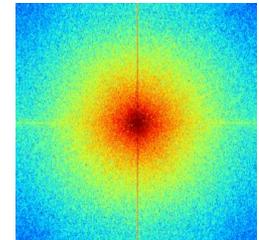
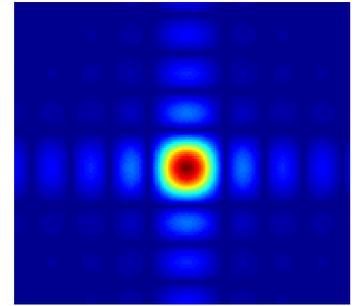
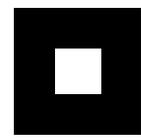
Perception

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 image of a spatial domain image shows the frequency components of that image. For example, a checkerboard pattern (A) would have two main peaks (3), a tulip field (B) would have a central peak with some structure (2), a scene with a light source (C) would have a central peak with a vertical line (4), a bright spot (D) would have a single central peak (1), and a scene with people and a horizon (E) would have a central peak with both horizontal and vertical lines (5).

Next class

- Denoising
- Template matching
- Image pyramids
- Introduce project 2

Questions