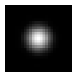
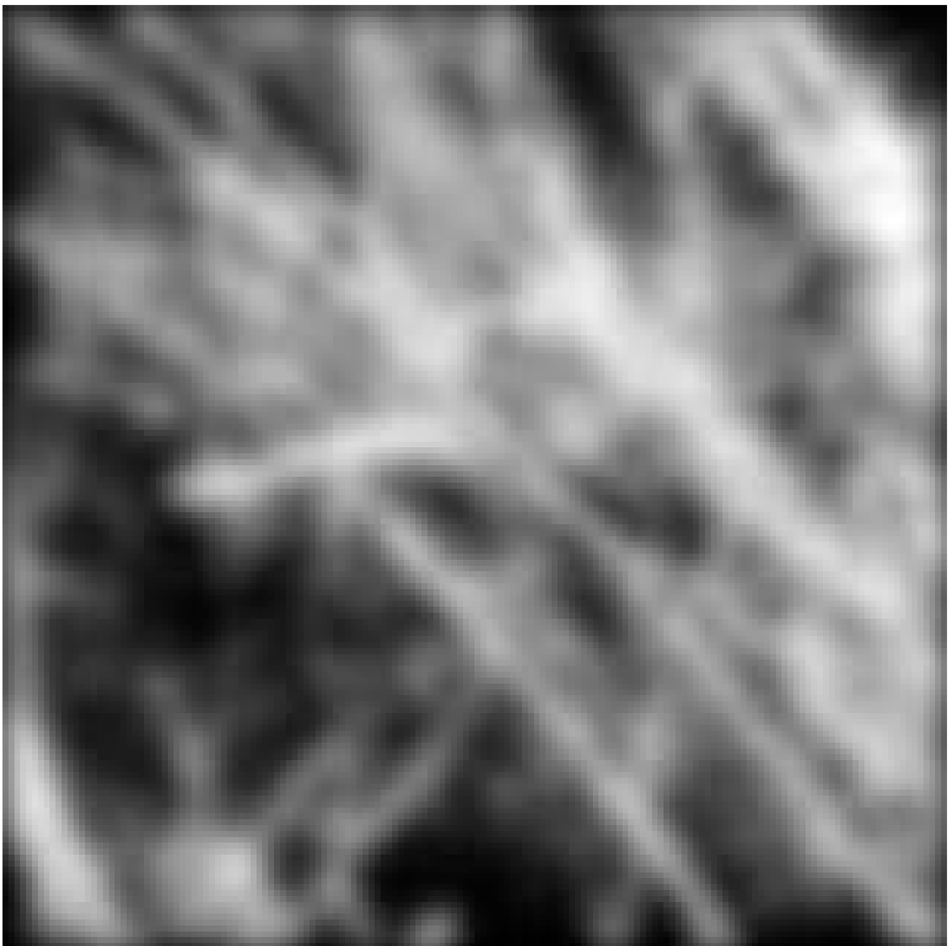
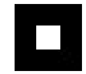
Thinking in Frequency

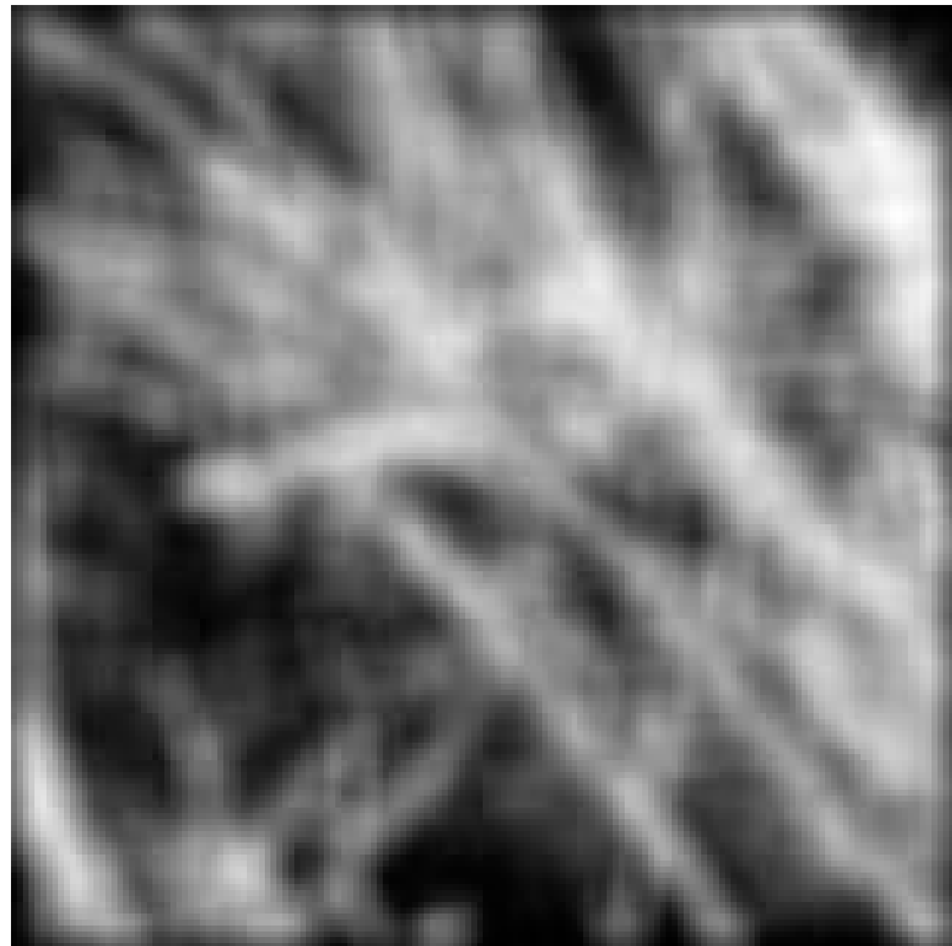
Slides adapted from James Hays, Derek Hoiem, Alyosha Efros, and Steven Lehar

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

Gaussian 



Box filter 



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

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

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

RGB image:

1 byte / sub-pixel

* 3 sub-pixels / pixel

* 4M pixels

12M bytes

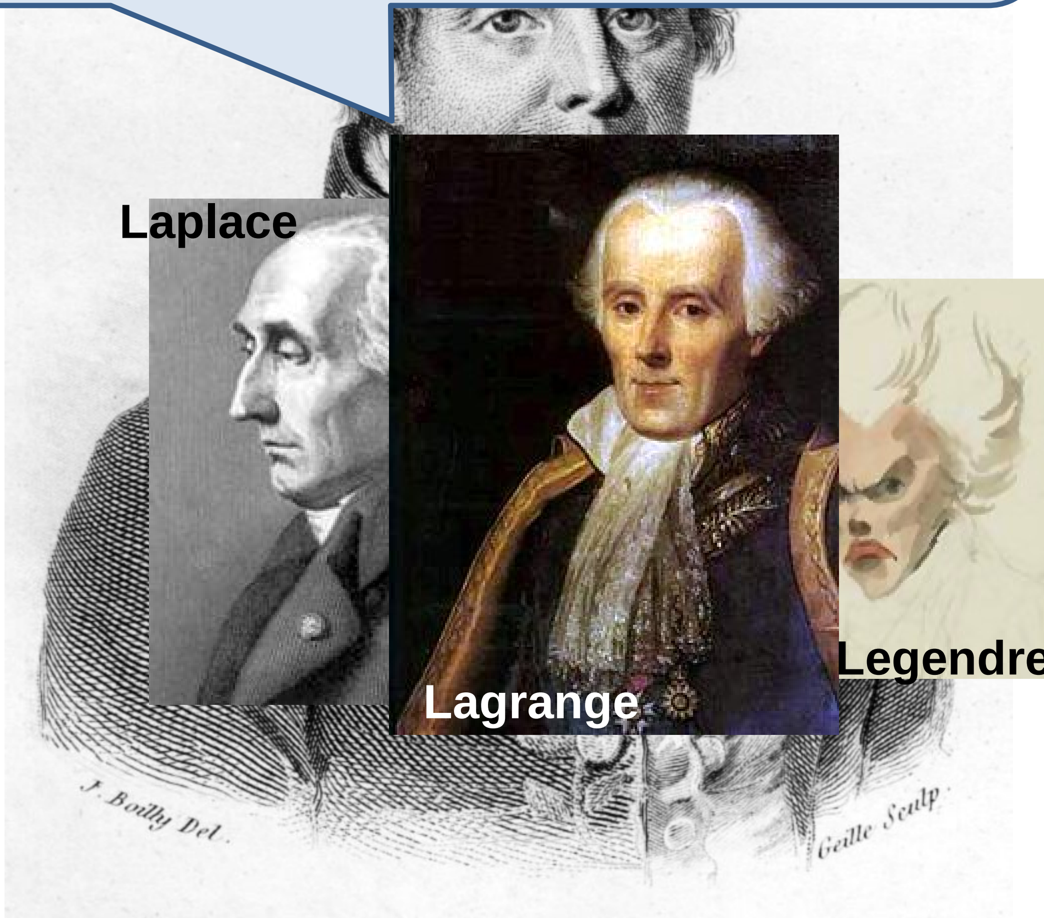
Jean Baptiste Joseph Fourier (1768-1830)

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.*

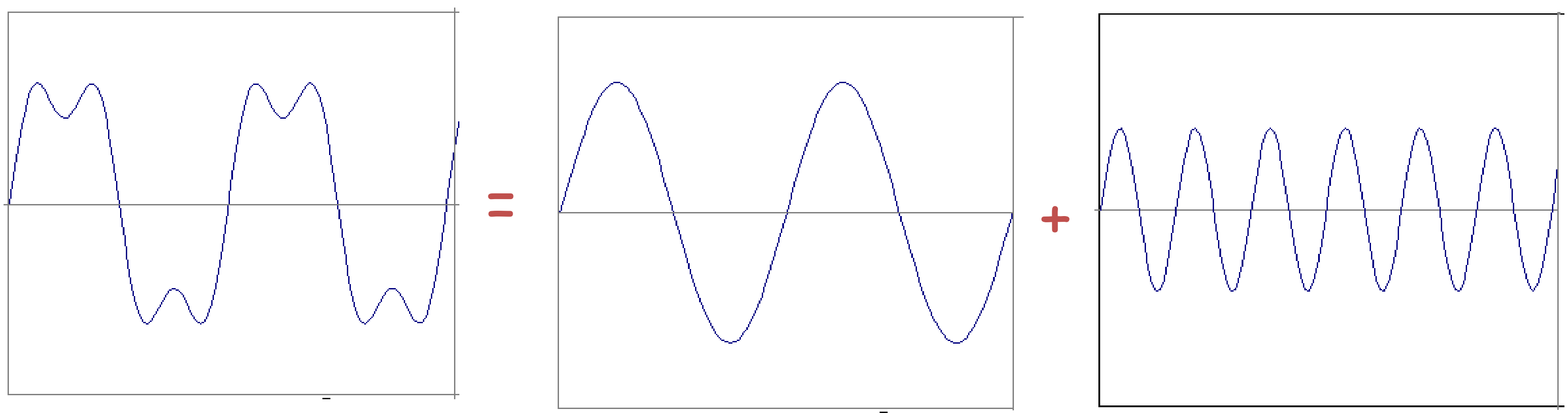
***Any** univariate function can*
rewritten as a weighted sum
sines and cosines of different
frequencies.

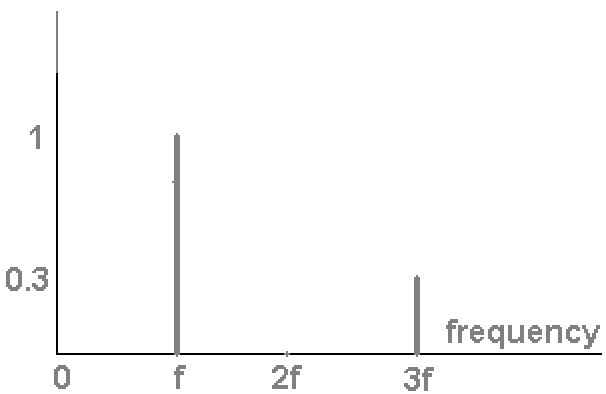
- Don't believe it?
  - Neither did Lagrange, Laplace, Poisson and other big wigs
  - Not translated into English until 1878!
- But it's (mostly) true!
  - called Fourier Series
  - there are some subtle restrictions



Frequency Spectra

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





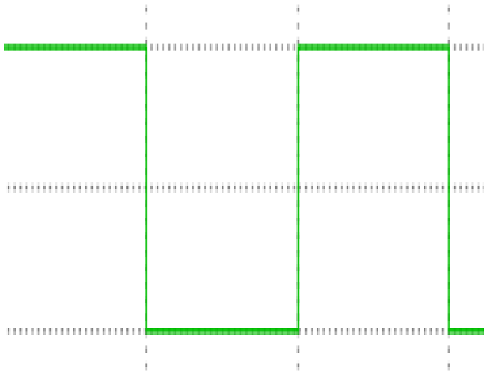
Our building block:

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

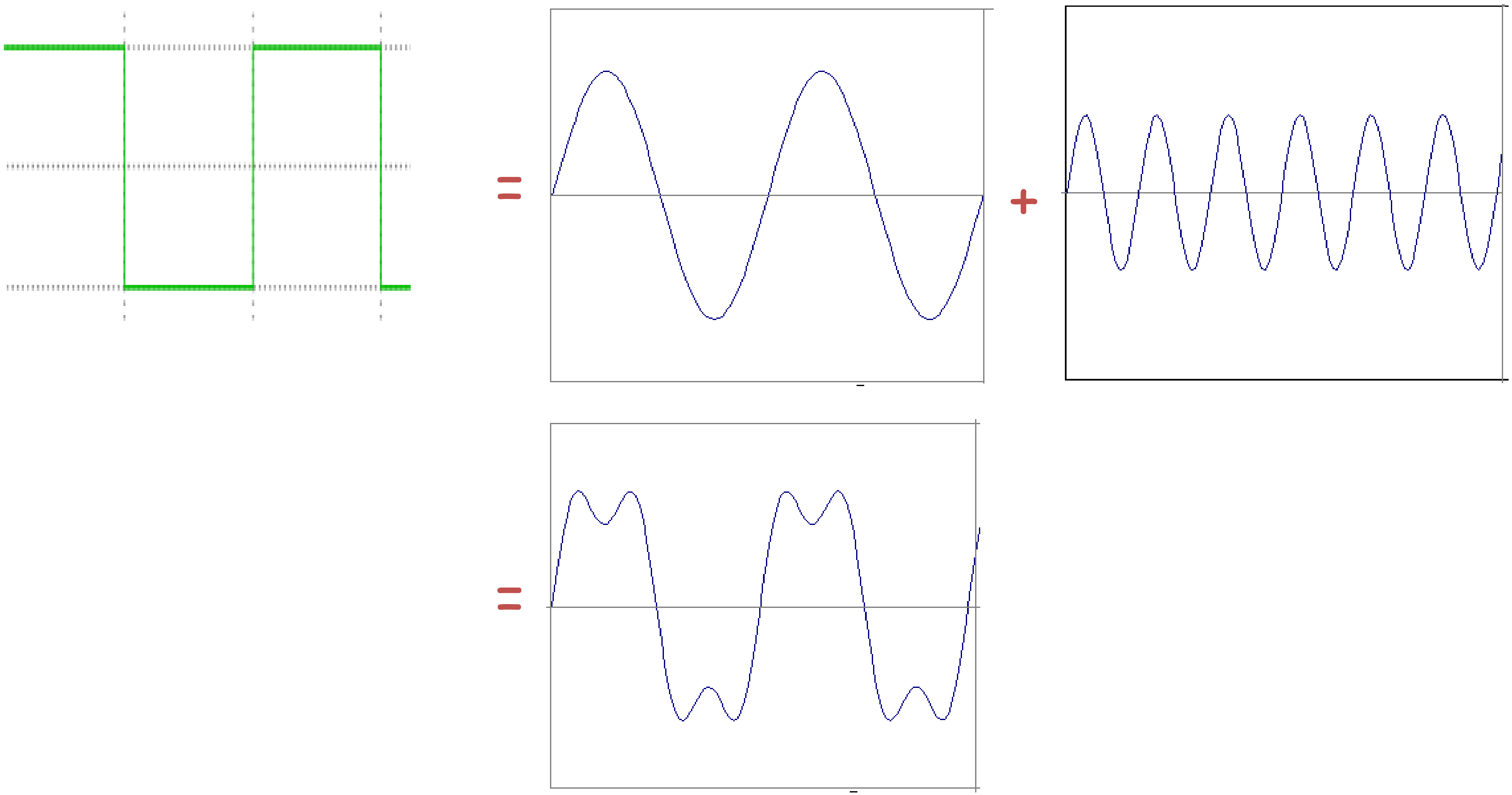
Add enough of them to get any signal *f(x)* you want!

Slides: Efros

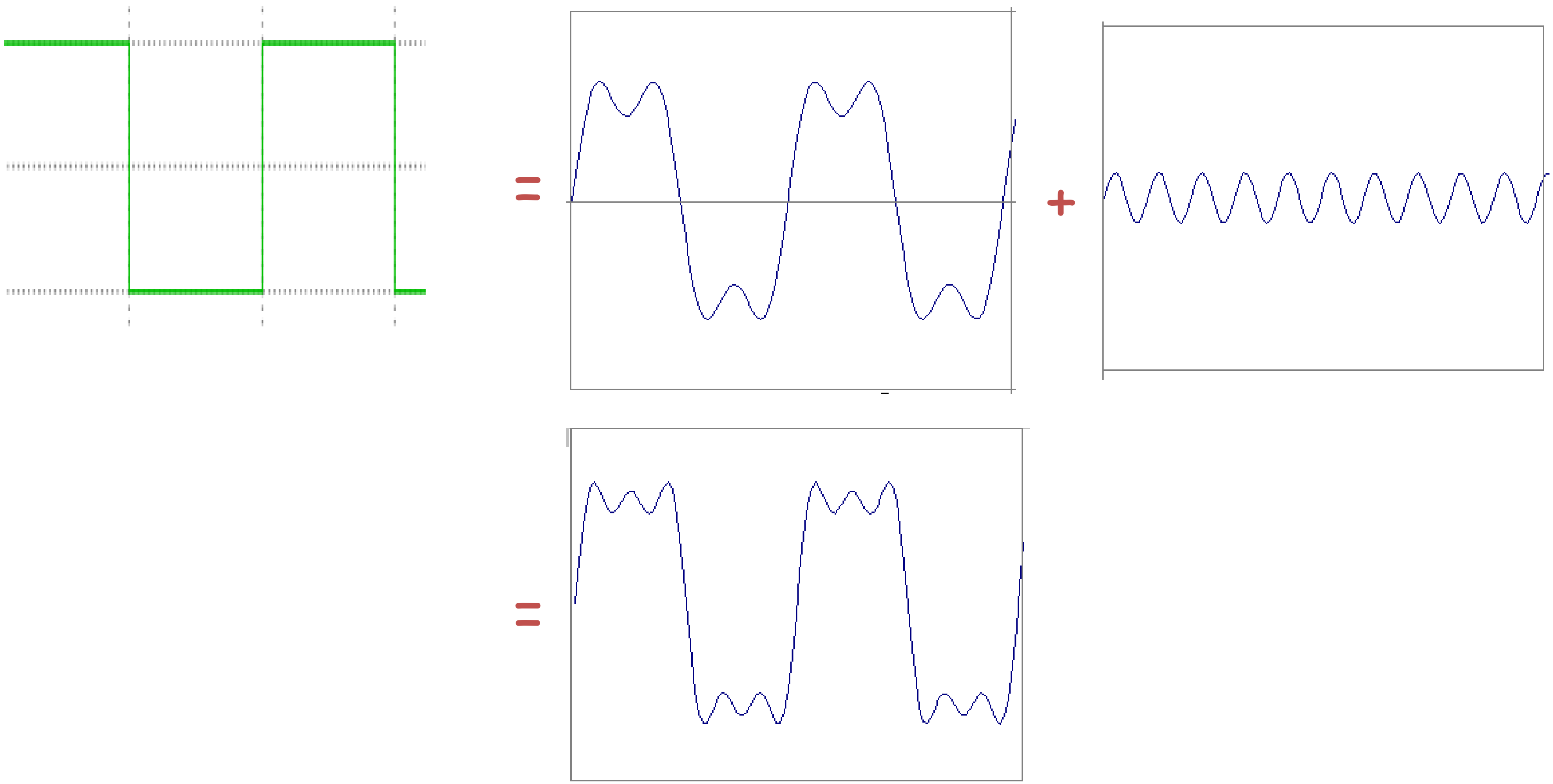
Frequency Spectra



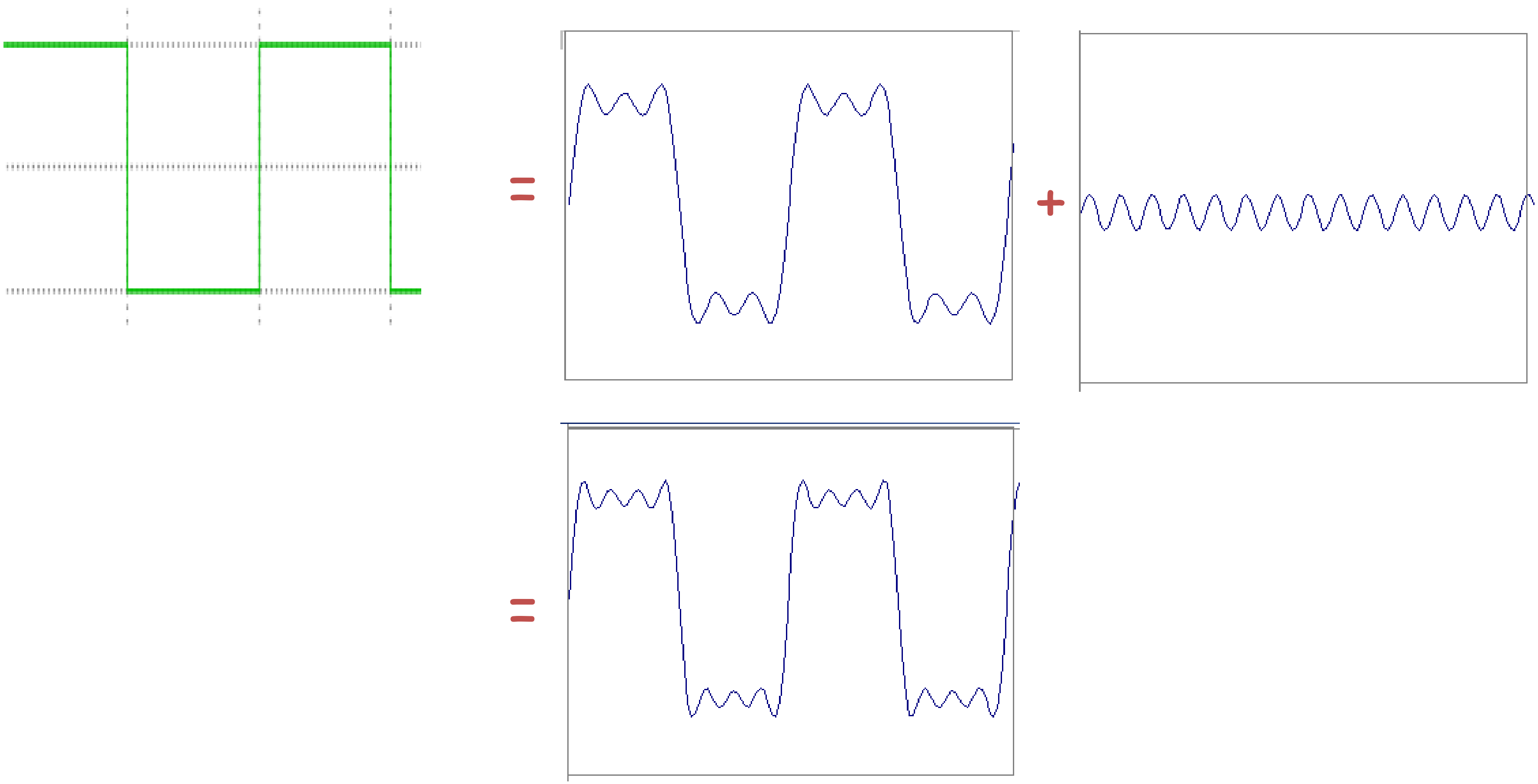
Frequency Spectra



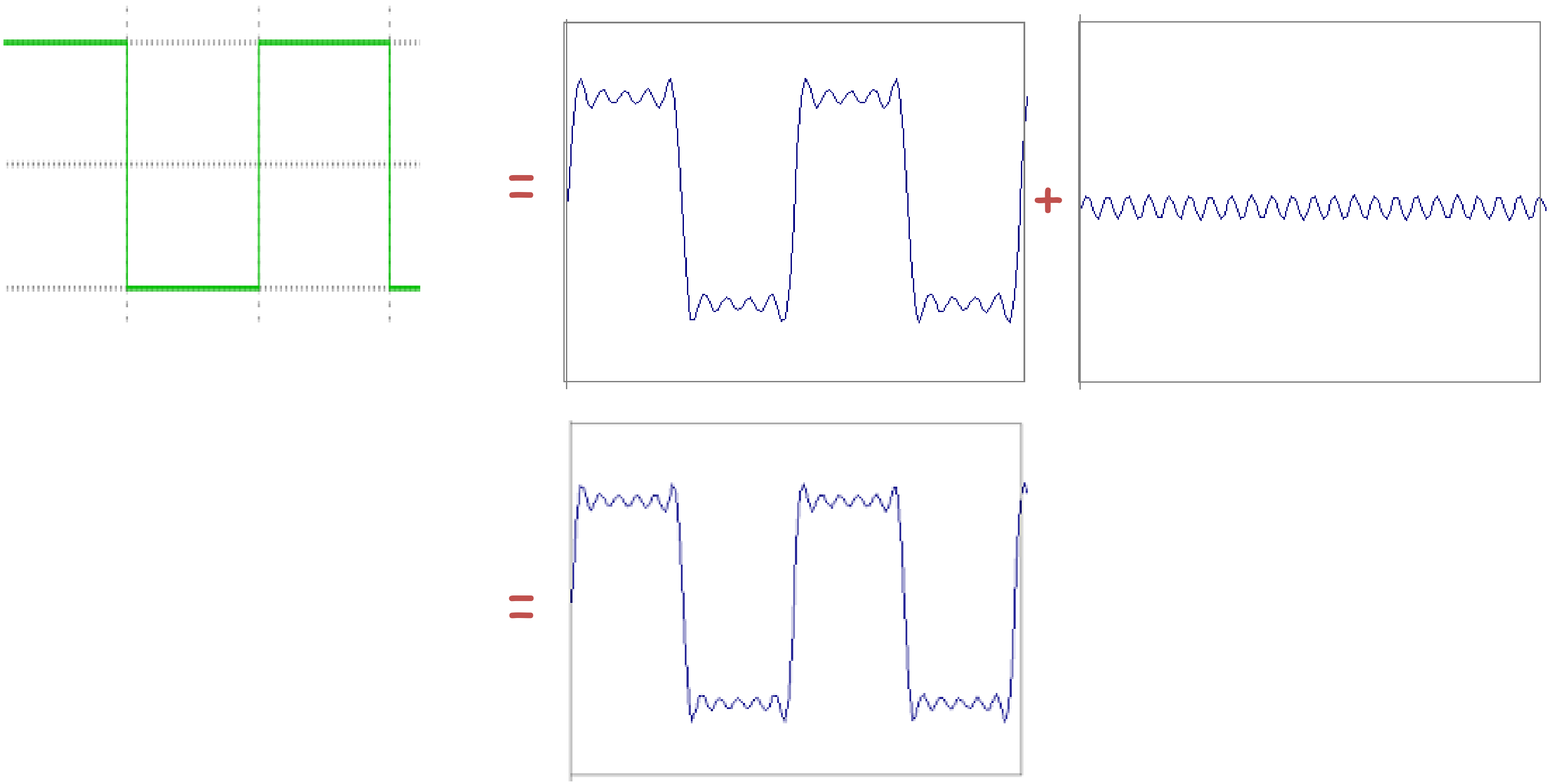
Frequency Spectra



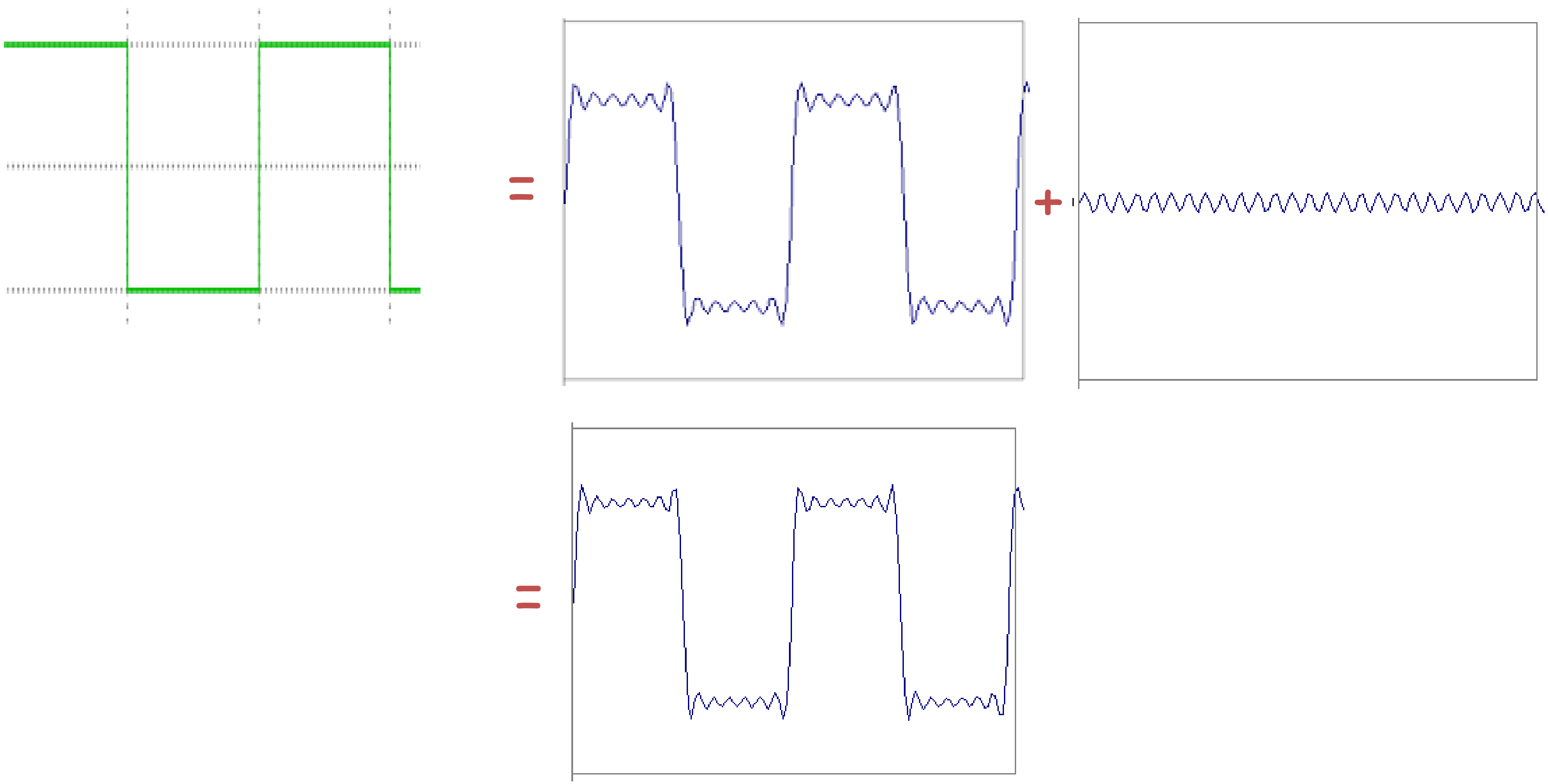
Frequency Spectra



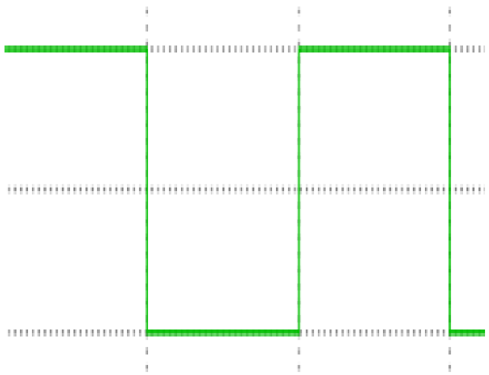
Frequency Spectra



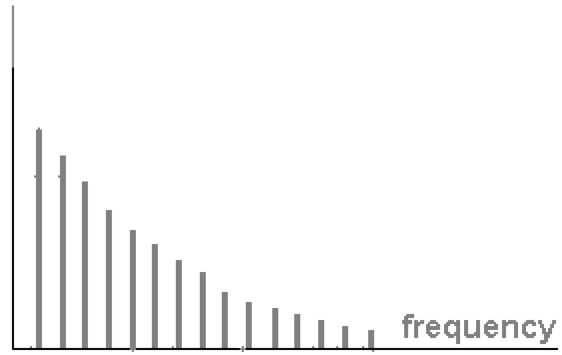
Frequency Spectra

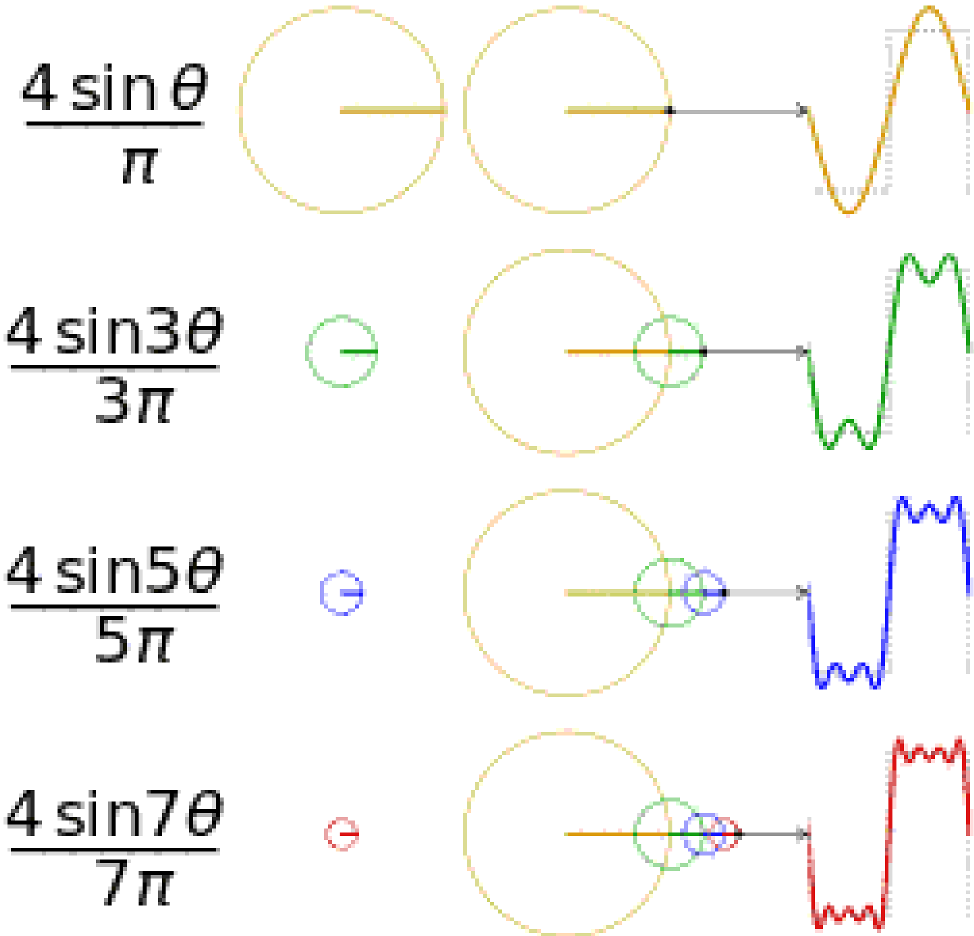


Frequency Spectra



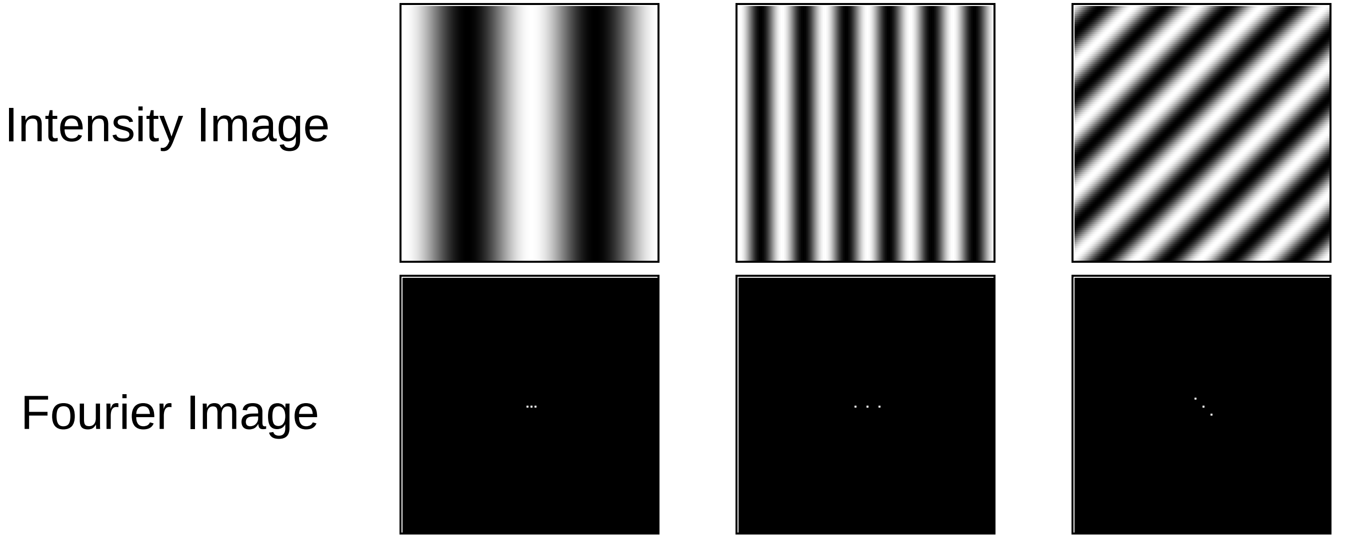
$$= \quad \boxed{A\sum_{k=1}^{\infty}\frac{1}{k}\sin(2\pi kt)}$$




$\frac{4\sin\theta}{\pi}$
$\frac{4\sin 3\theta}{3\pi}$
$\frac{4\sin 5\theta}{5\pi}$
$\frac{4\sin 7\theta}{7\pi}$

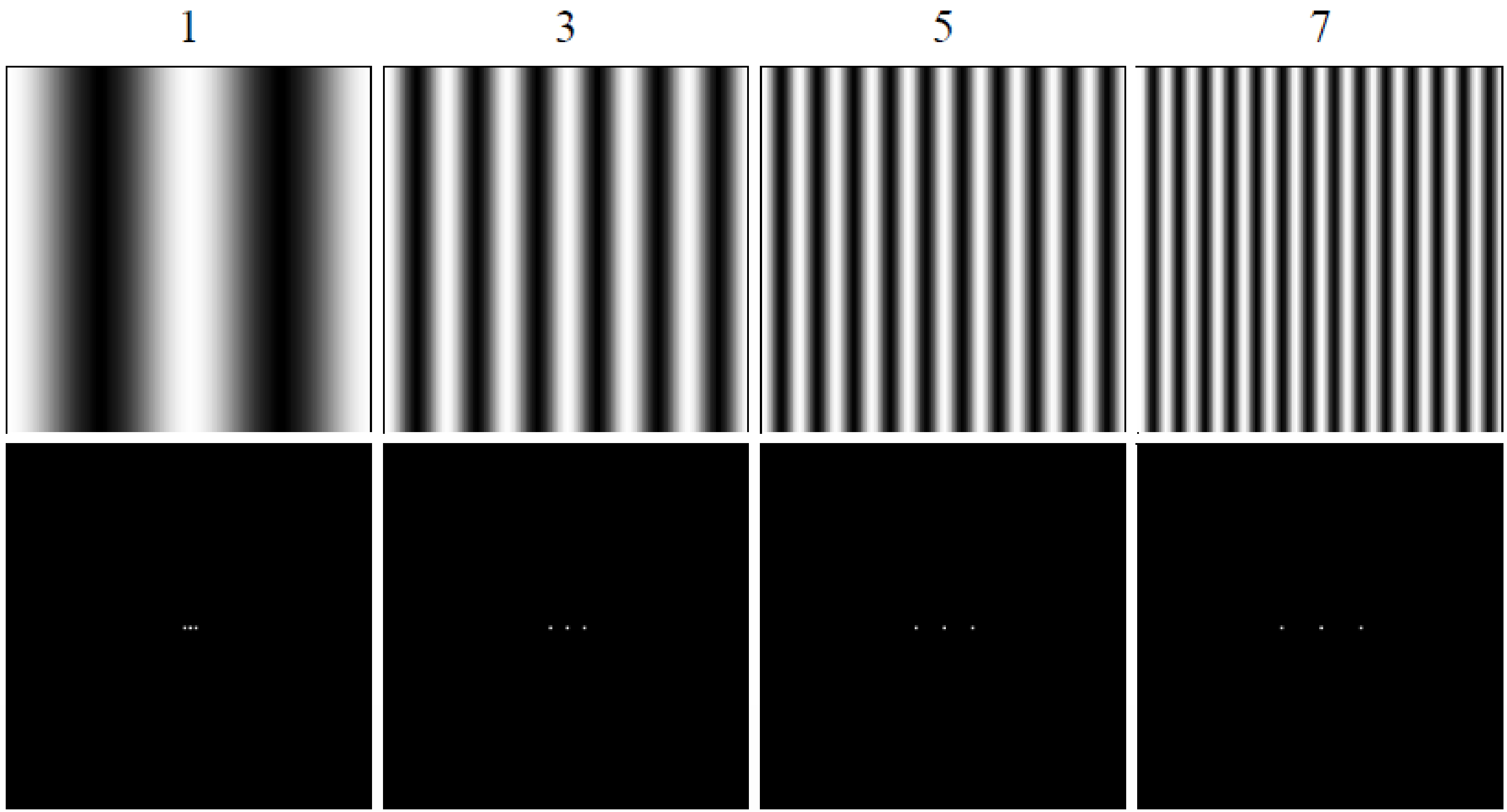
Fourier analysis in images

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

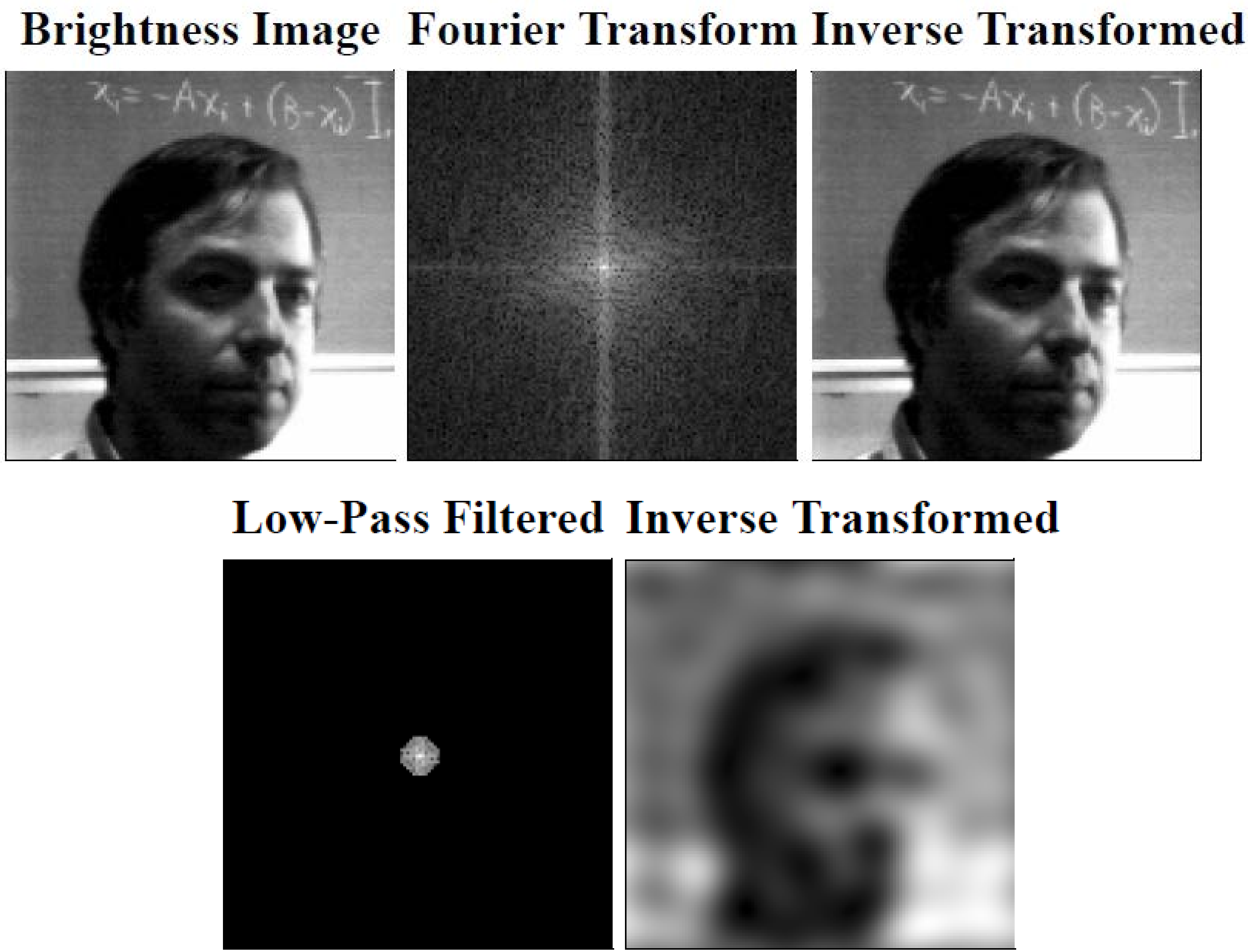


http://sharp.bu.edu/~slehar/fourier/fourier.html#filtering

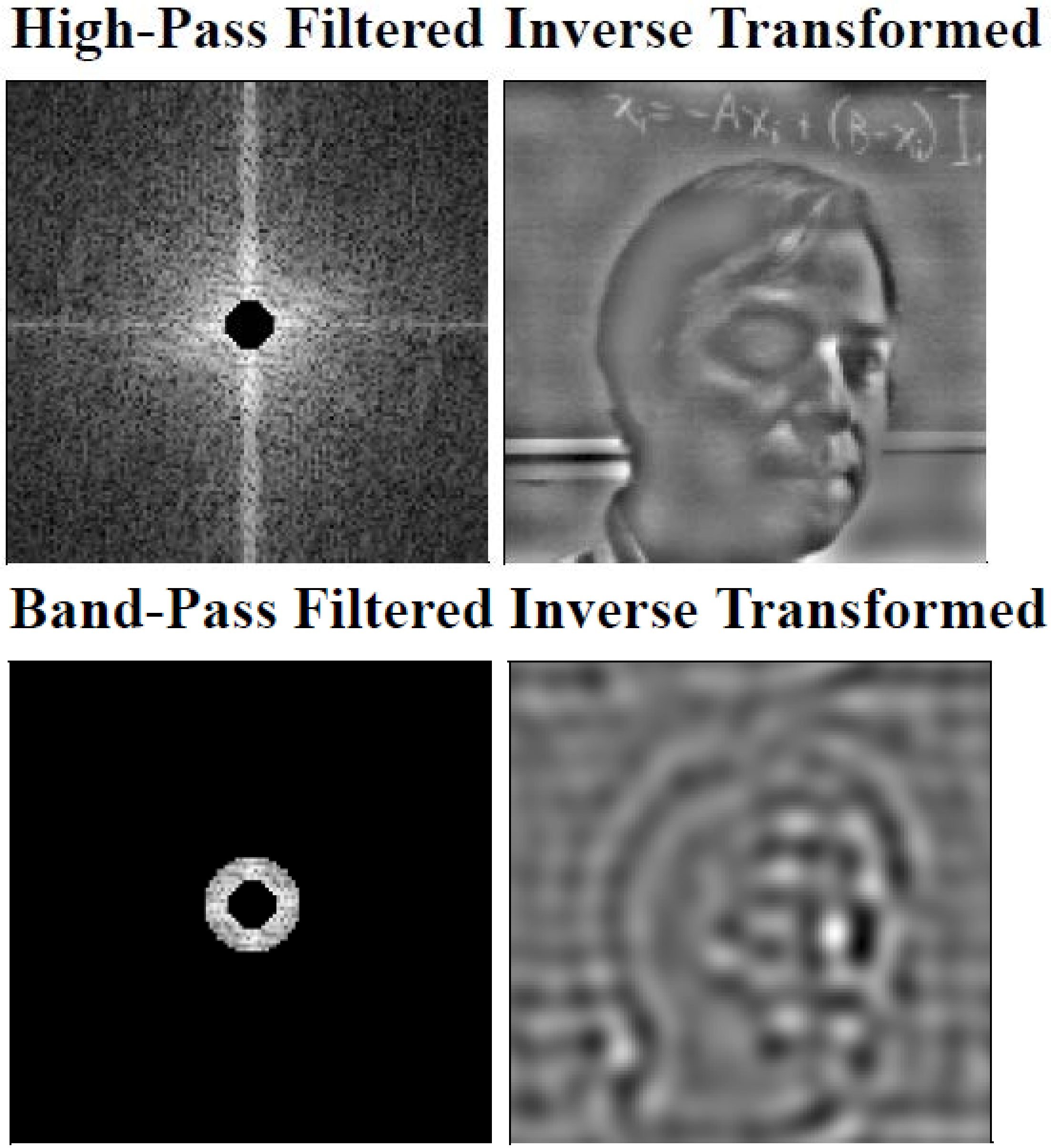
Frequency <-> Spatial



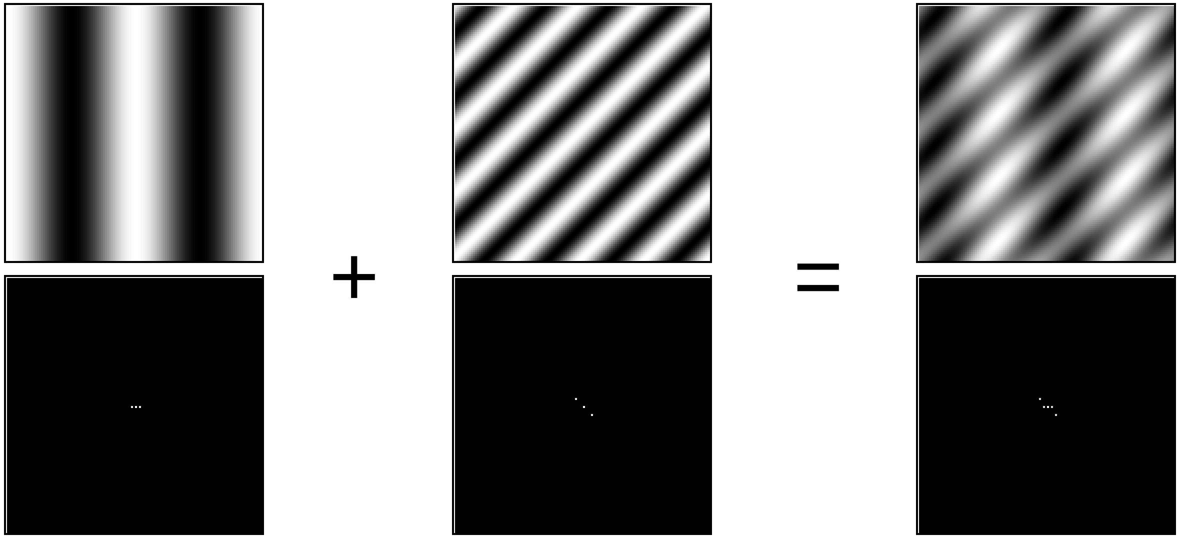
Frequency <-> Spatial



Frequency <-> Spatial



Signals can be composed



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

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

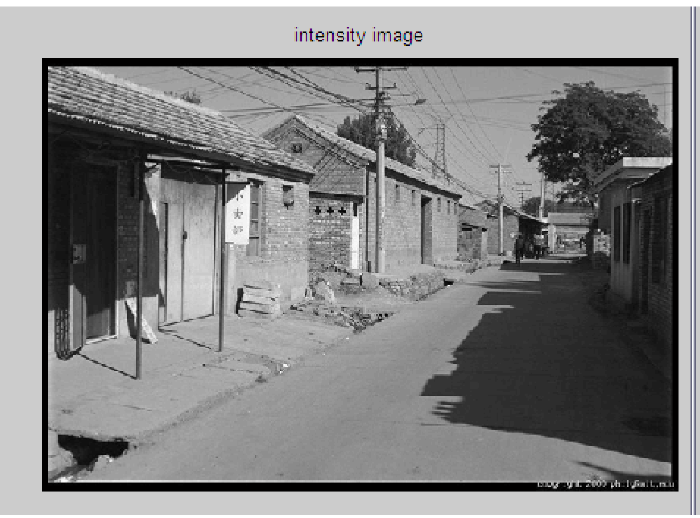
- Fourier transform of a real signal is symmetric about the origin

- Can be computed efficiently
  - Fast Fourier Transform (FFT): O (N log N)

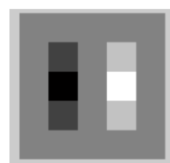
See Szeliski Book (3.4)

Filtering in spatial domain

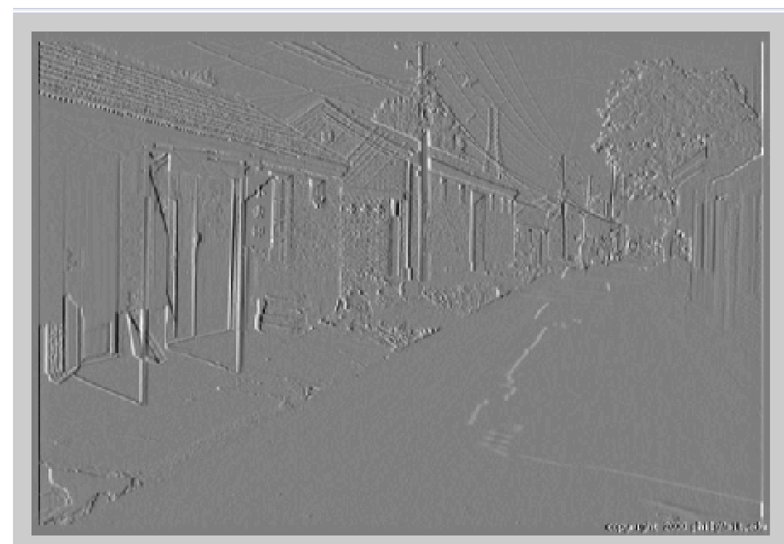
1	0	-1
2	0	-2
1	0	-1



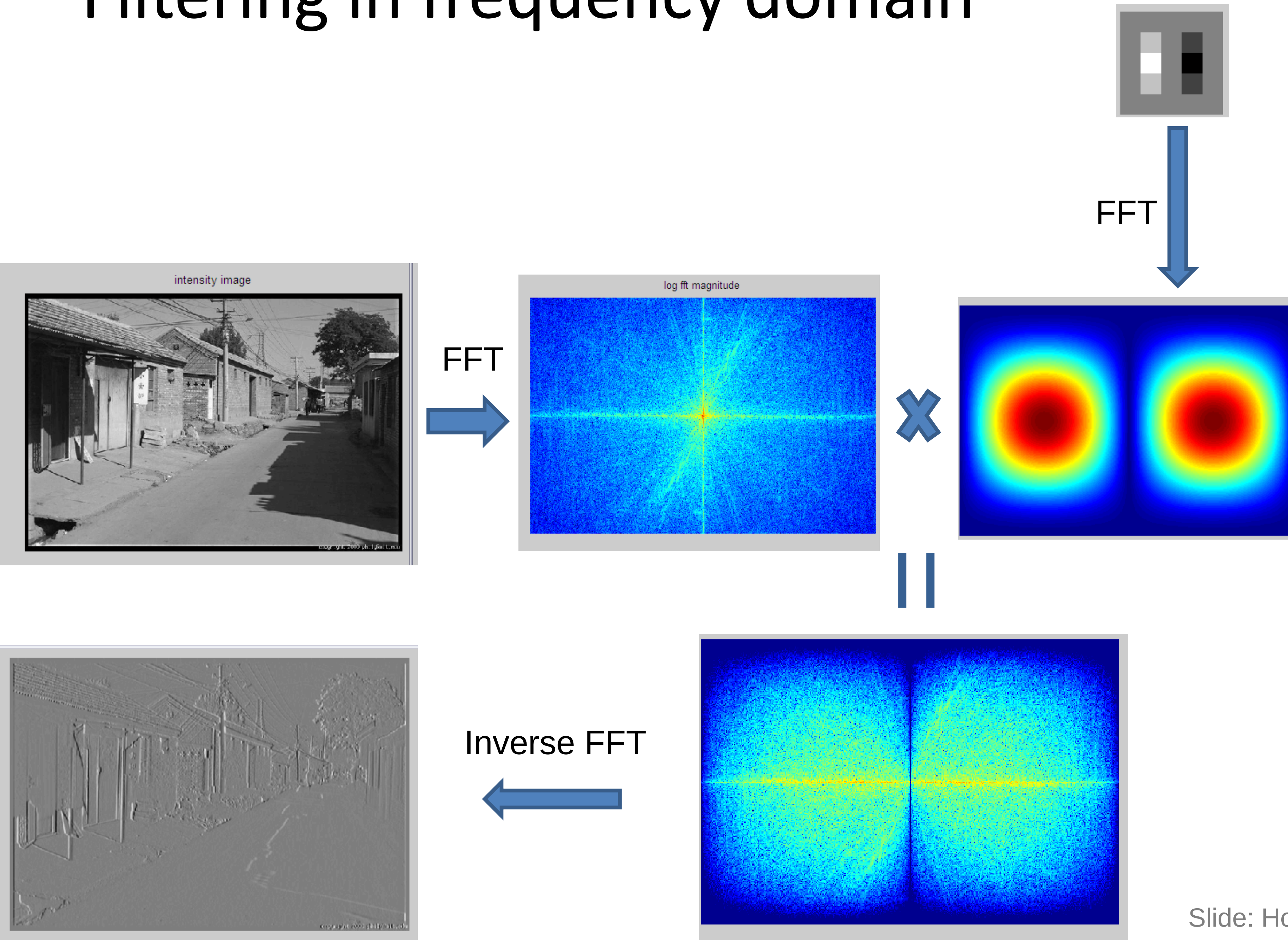
*



=



Filtering in frequency domain



Slide: Hoiem

FFT in Matlab

- Filtering with fft

```
im = double(imread('…'))/255;
im = rgb2gray(im); % "im" should be a gray-scale floating point image
[imh, imw] = size(im);

hs = 50; % filter half-size
fil = fspecial('gaussian', hs*2+1, 10);

fftsize = 1024; % should be order of 2 (for speed) and include padding
im_fft  = fft2(im,  fftsize, fftsize);                          % 1) fft im with padding
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
```

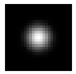
- Displaying with fft

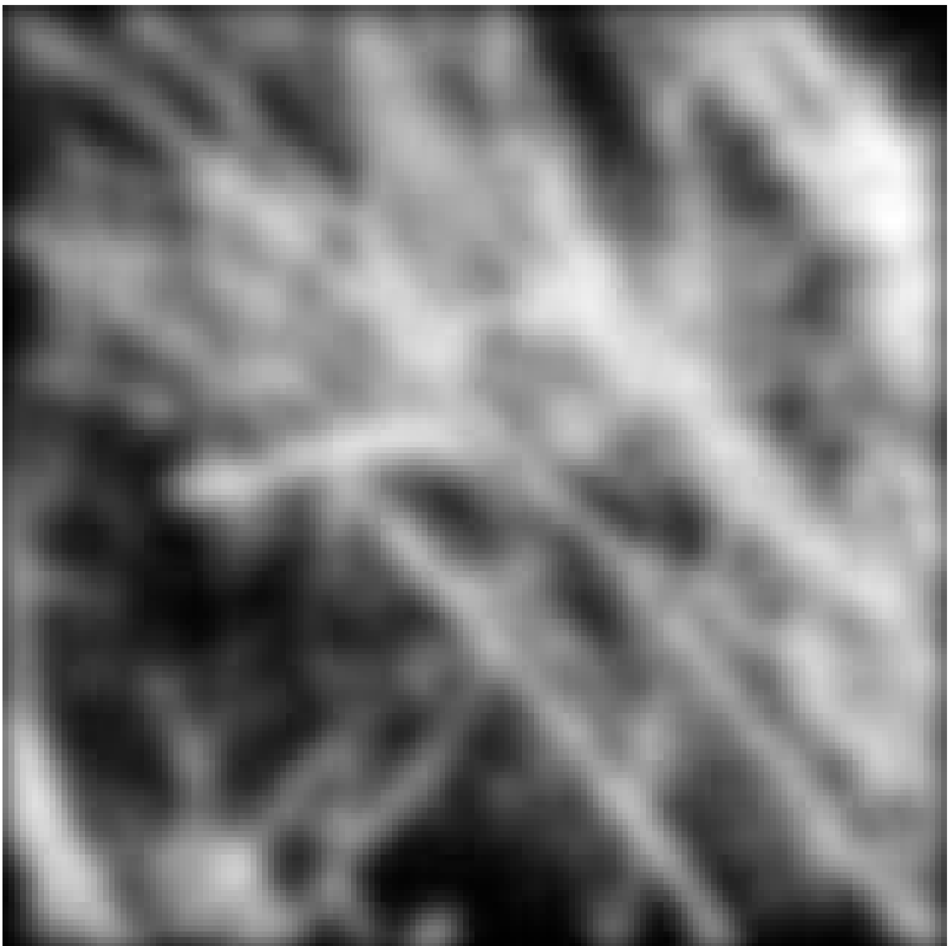
```
figure(1), imagesc(log(abs(fftshift(im_fft)))), axis image, colormap jet
```

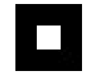
Slide: Hoiem

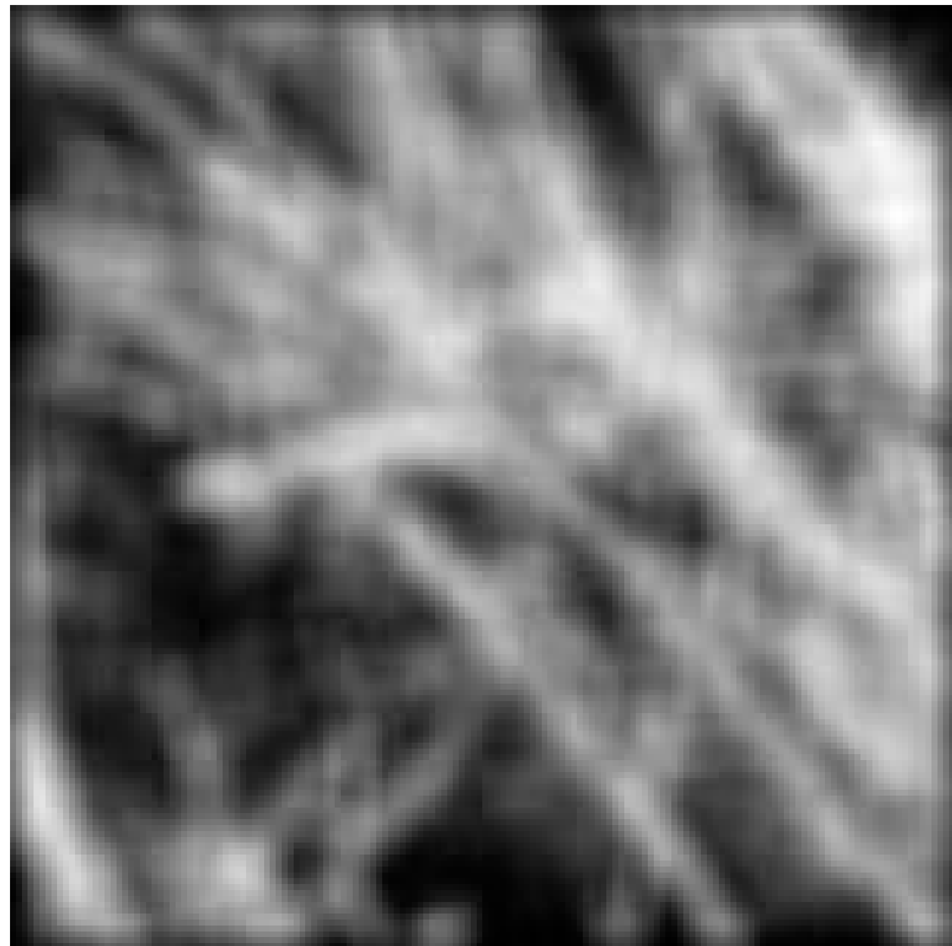
Filtering

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

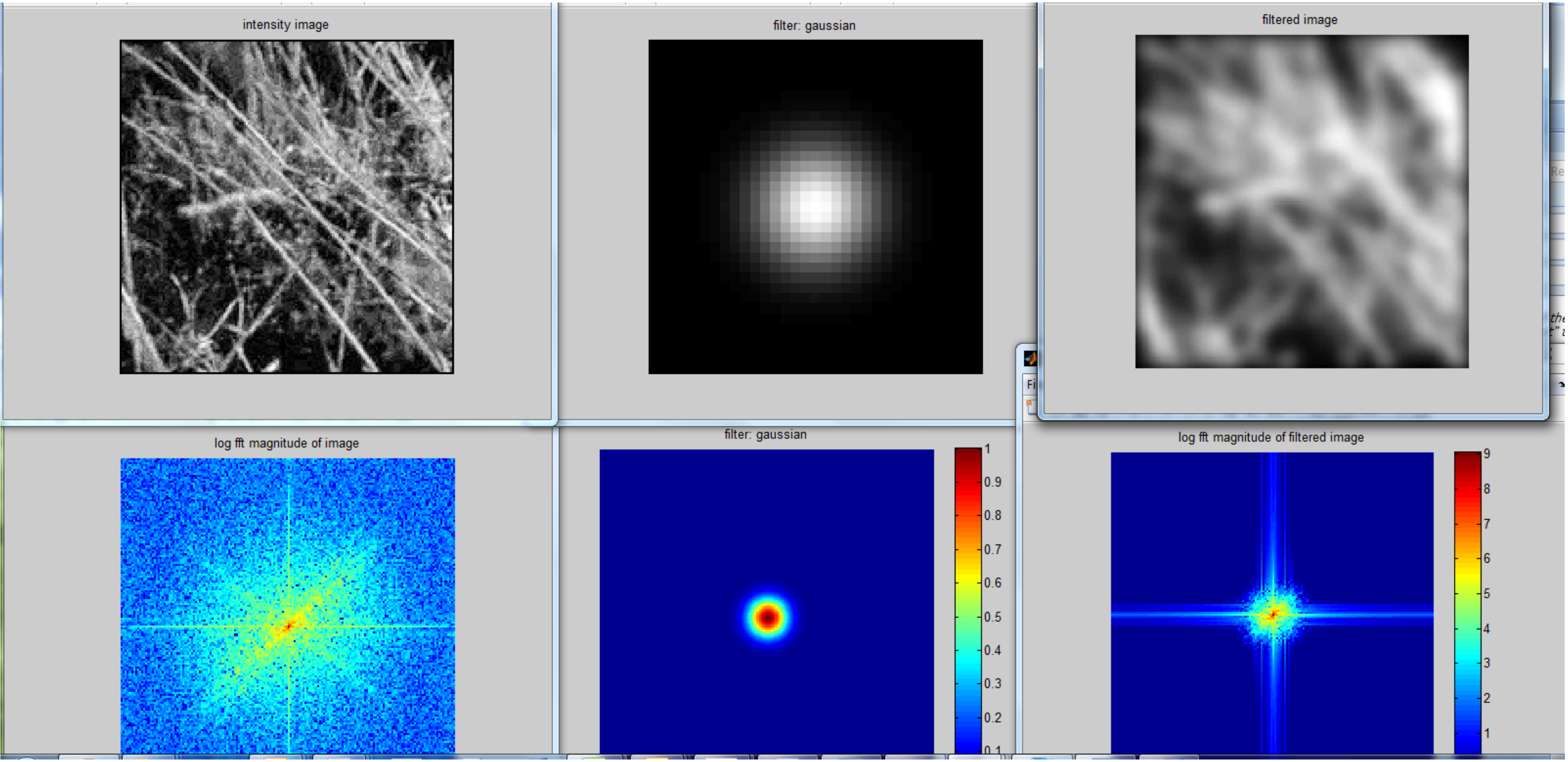
Gaussian 



Box filter 

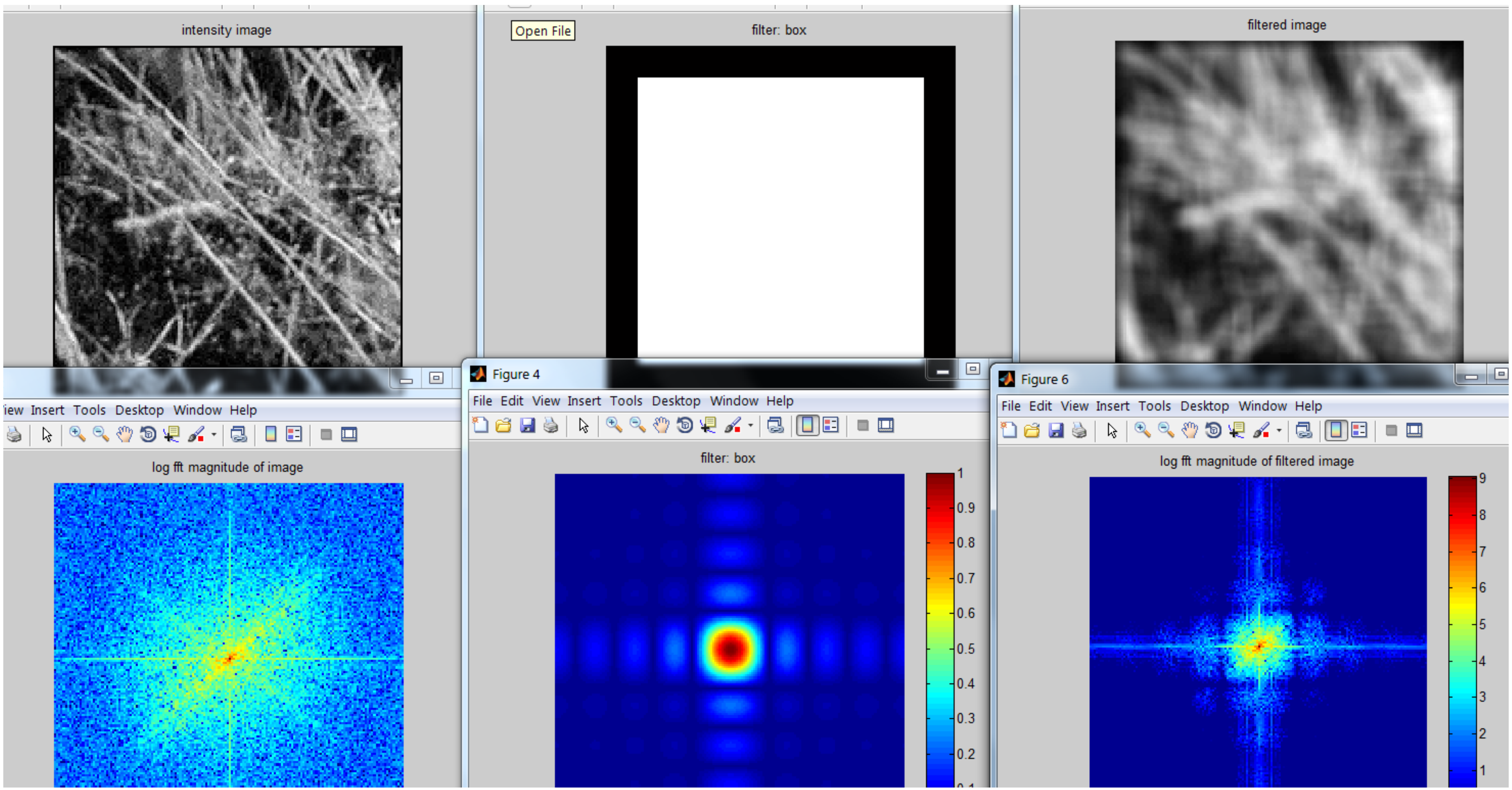


Gaussian



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



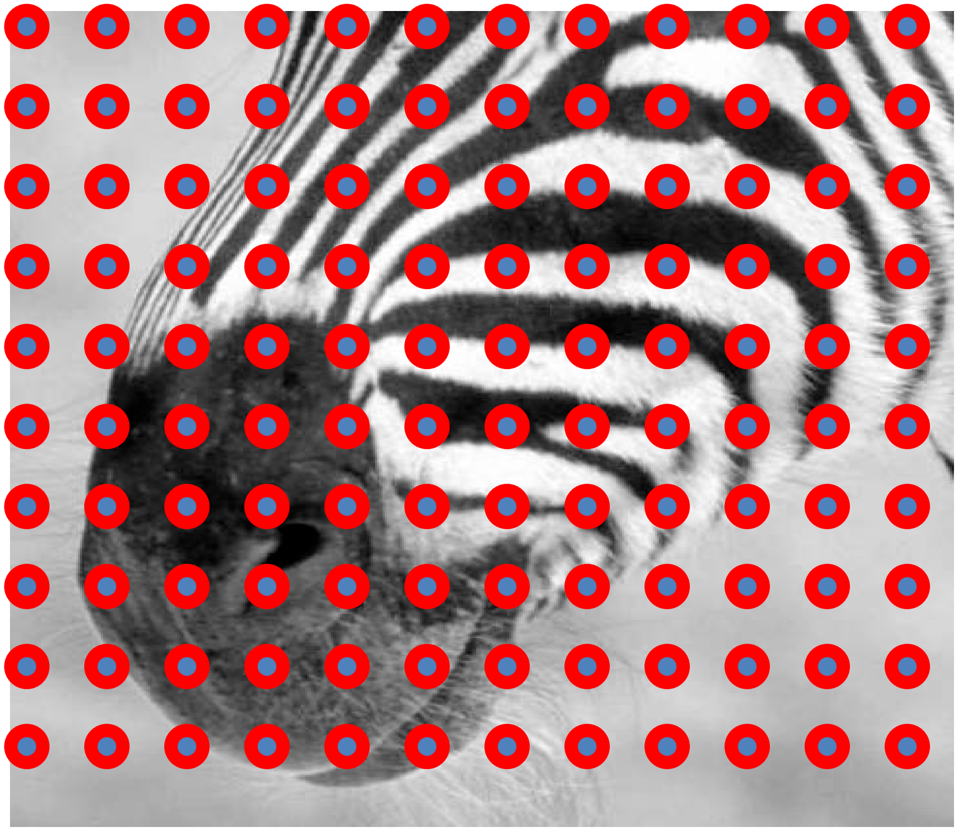
intensity image
Open File
filter: box
filtered image
Figure 4
Figure 6
File Edit View Insert Tools Desktop Window Help
log fft magnitude of image
filter: box
log fft magnitude of filtered image

Sampling

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

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

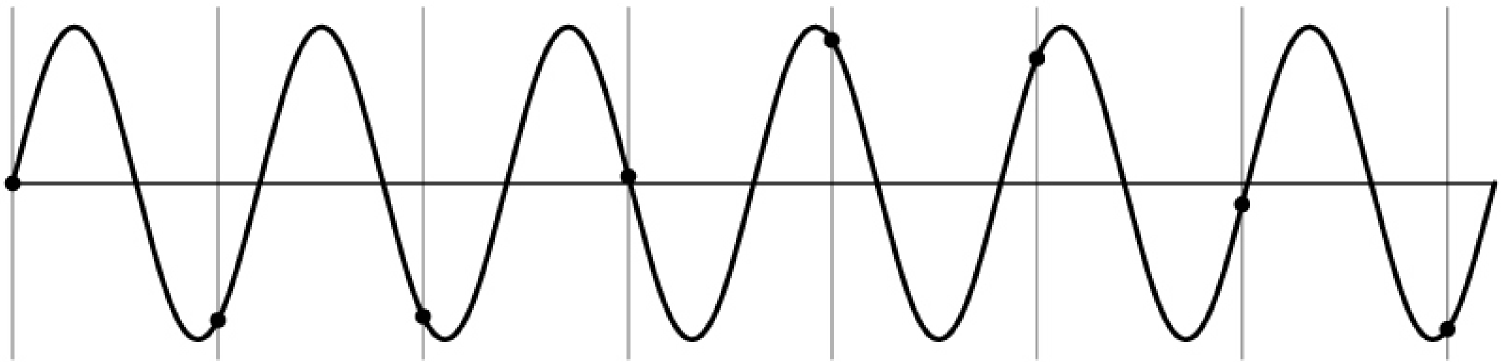
Subsampling by a factor of 2



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

Aliasing problem

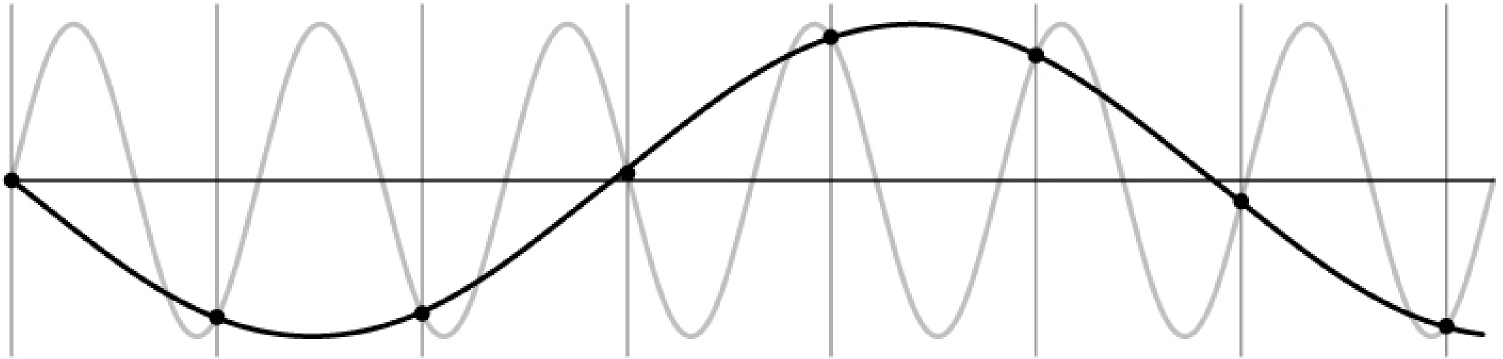
- 1D example (sinewave):



Source: S. Marschner

Aliasing problem

- 1D example (sinewave):



Source: S. Marschner

Aliasing problem

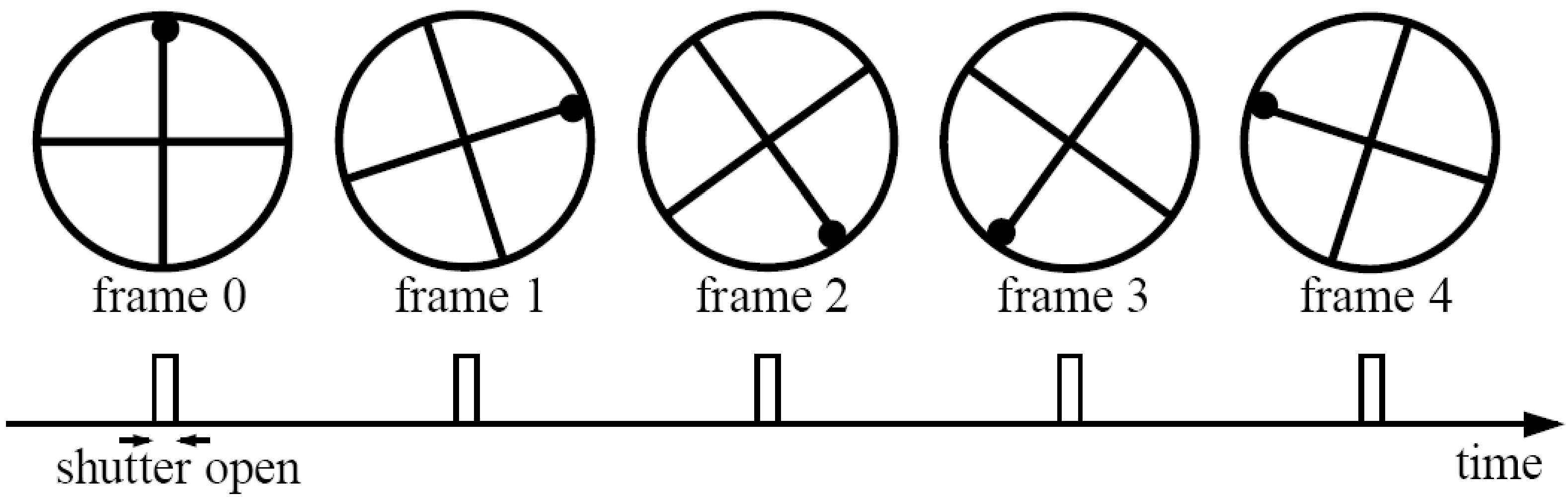
- Sub-sampling may be dangerous….
- Characteristic errors may appear:
  - “Wagon wheels rolling the wrong way in movies”
  - “Checkerboards disintegrate in ray tracing”

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)

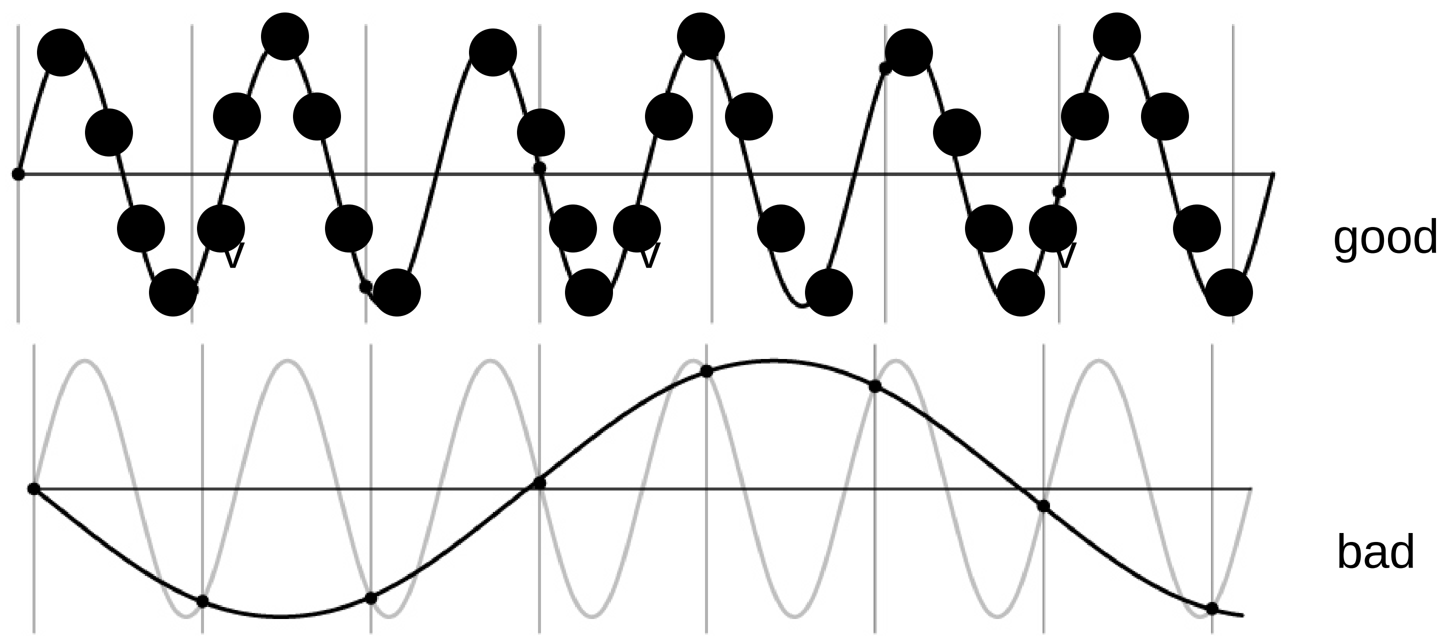
Slide by Steve Seitz

Aliasing in graphics

Source: A. Efros

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

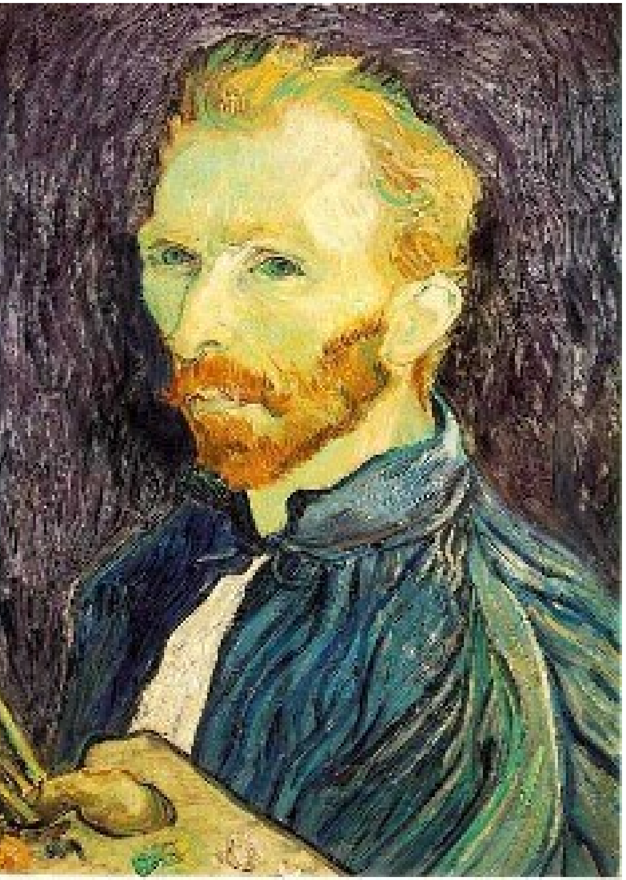
1. Start with image(h, w)
2. Apply low-pass filter

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

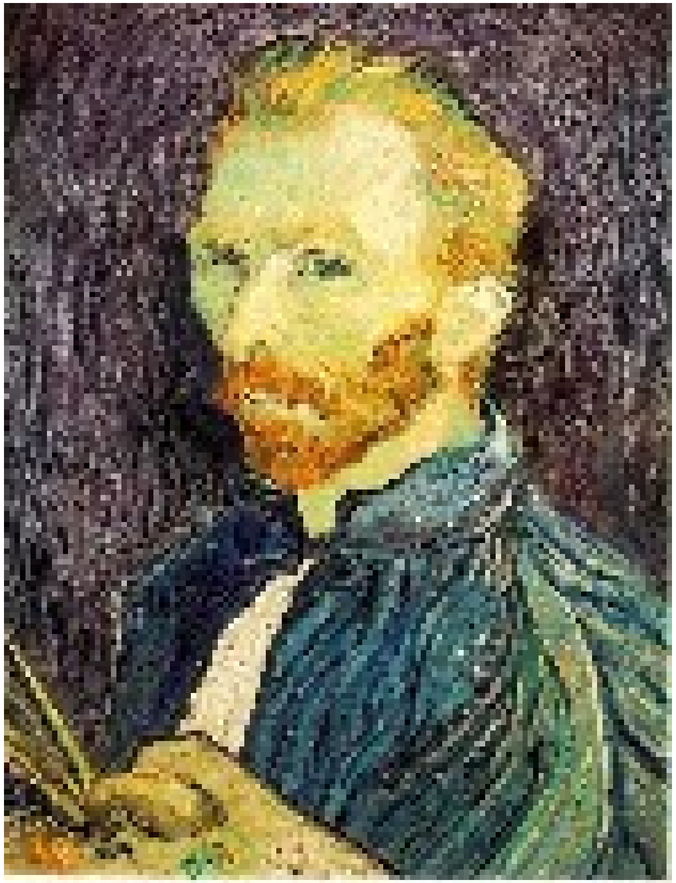
3. Sample every other pixel

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

Subsampling without pre-filtering



1/2

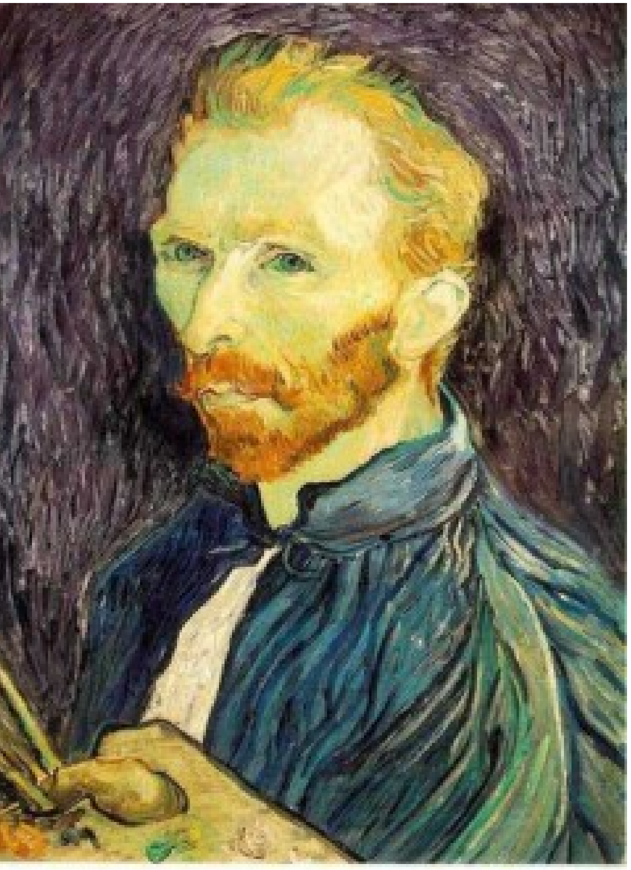


1/4 (2x zoom)

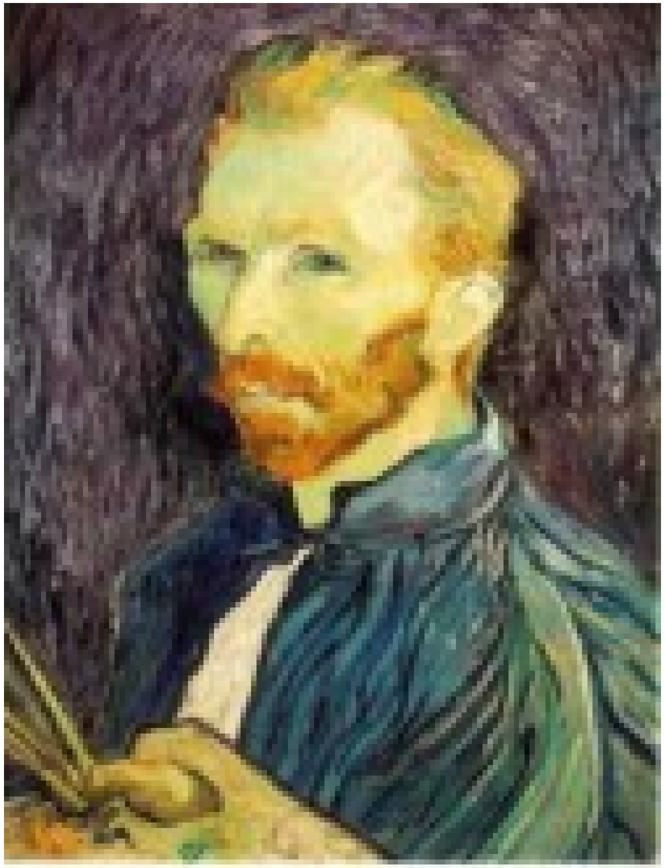
1/8 (4x zoom)

Slide by Steve Seitz

Subsampling with Gaussian pre-filtering



Gaussian 1/2

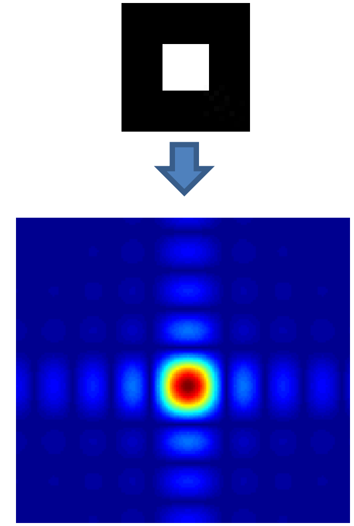


G 1/4

G 1/8

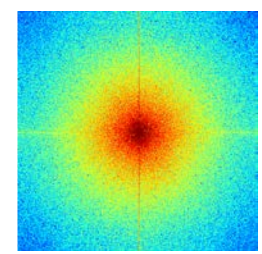
Slide by Steve Seitz

Things to Remember



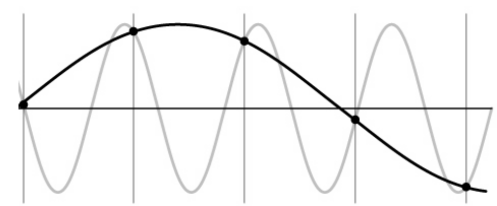
- Sometimes it makes sense to think of images and filtering in the frequency domain
  - Fourier analysis

- 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

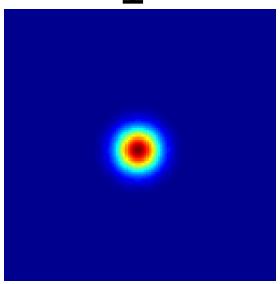
- Remember to low-pass before sampling



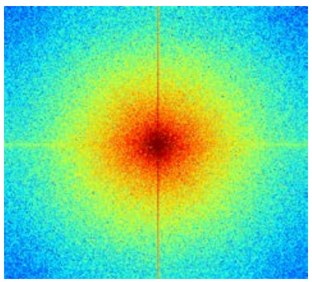
Practice question

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

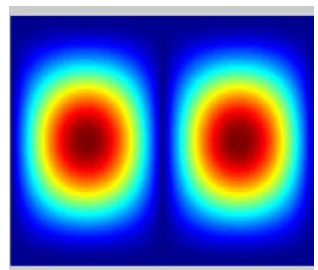
1



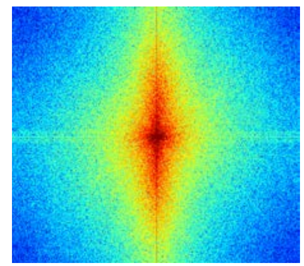
2



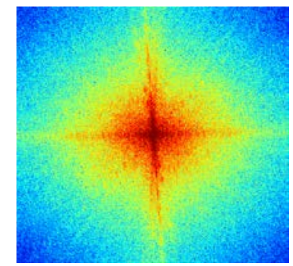
3



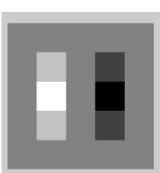
4



5

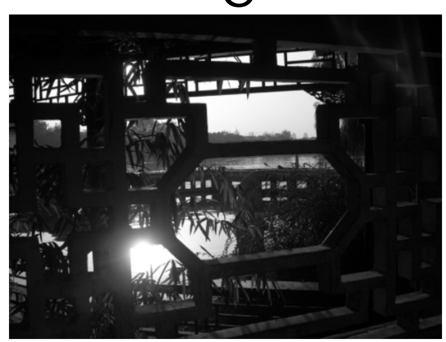


A

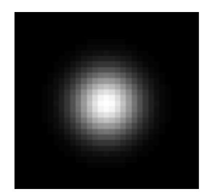


B

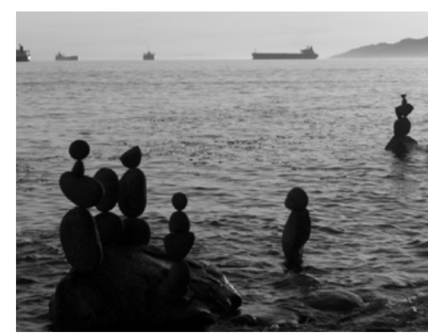
C



D



E



Next class

- Template matching
- Image Pyramids
- Filter banks and texture
- Denoising, Compression