Lecture 6: Image Processing (cont.)

COMP 590/776: Computer Vision

Instructor: Soumyadip (Roni) Sengupta

TA: Mykhailo (Misha) Shvets

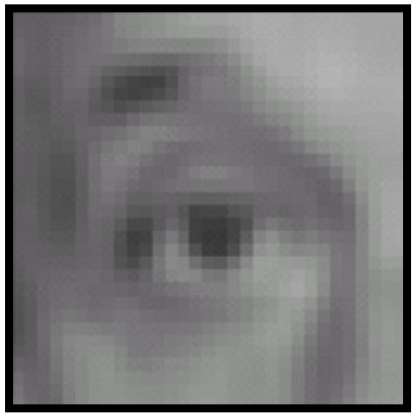
Course Website:
Scan Me!

Recap

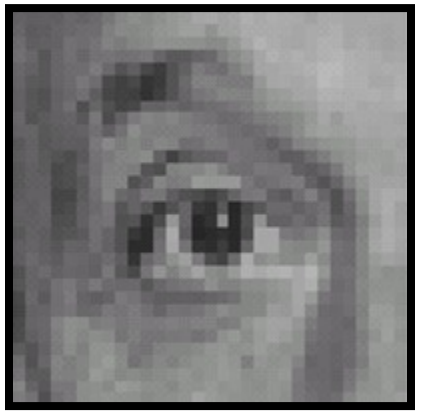
Filters

Many Options:

- Mean Filtering
- Gaussian Filtering
- Bilateral Filtering



Blur



Original

Linear Filtering:
Cross-correlation
& Convolution

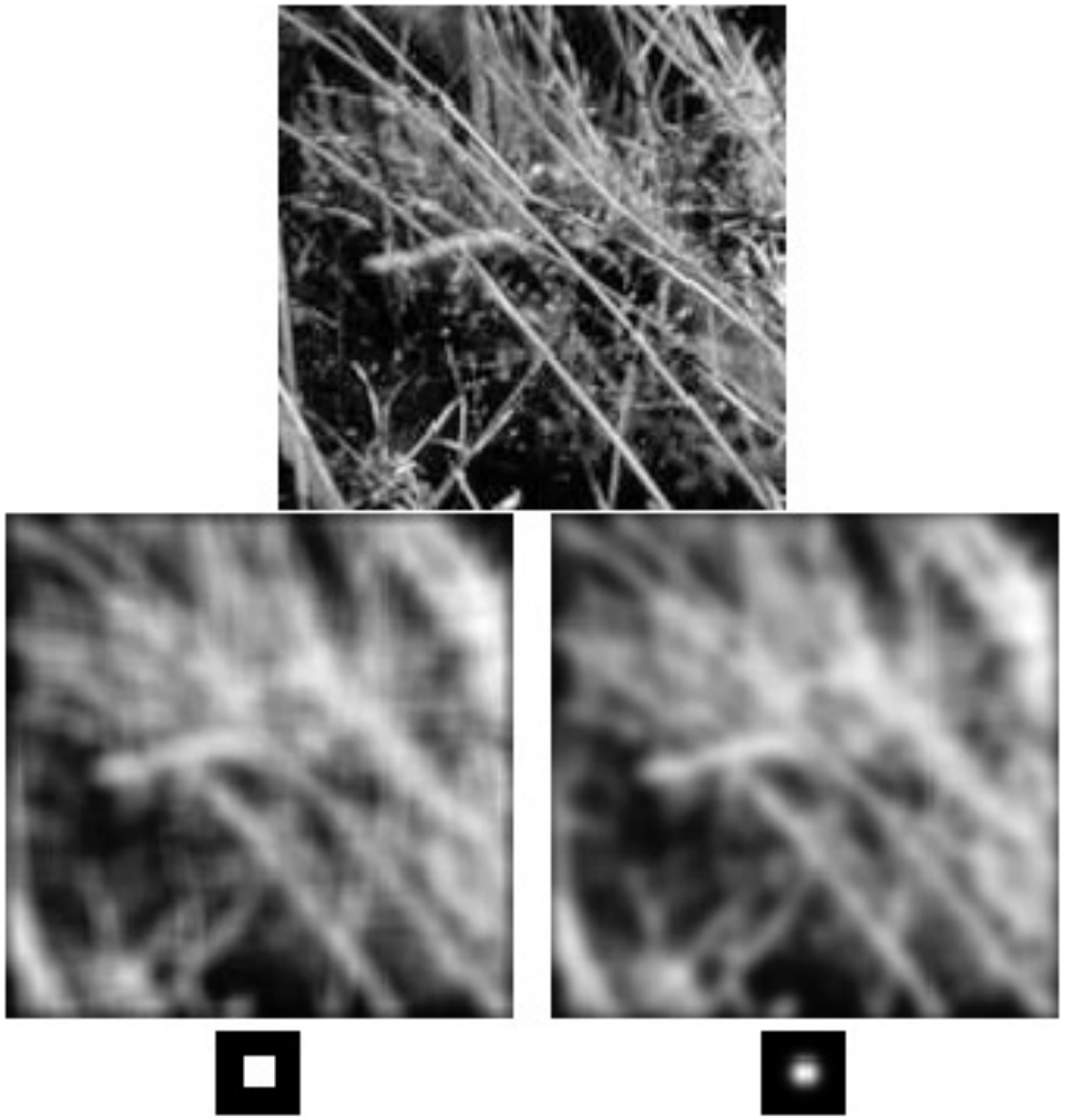
Shifted left by 1 pixel

$$*\begin{bmatrix}0&0&0\\1&0&0\\0&0&0\end{bmatrix}$$

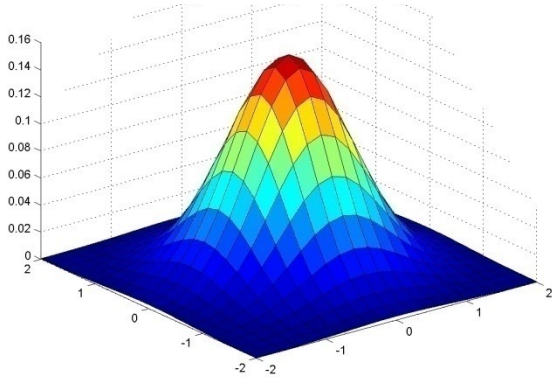
Sharpening

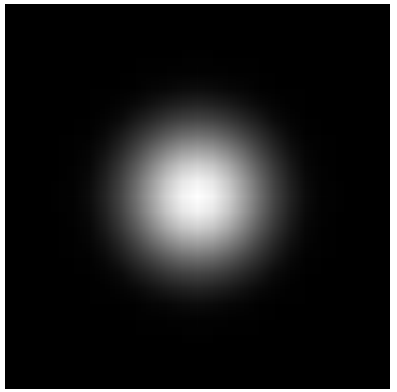
Laplacian Filtering

Mean vs. Gaussian filtering



$$G_\sigma = \frac{1}{2\pi\sigma^2} e^{-\frac{(x^2+y^2)}{2\sigma^2}}$$





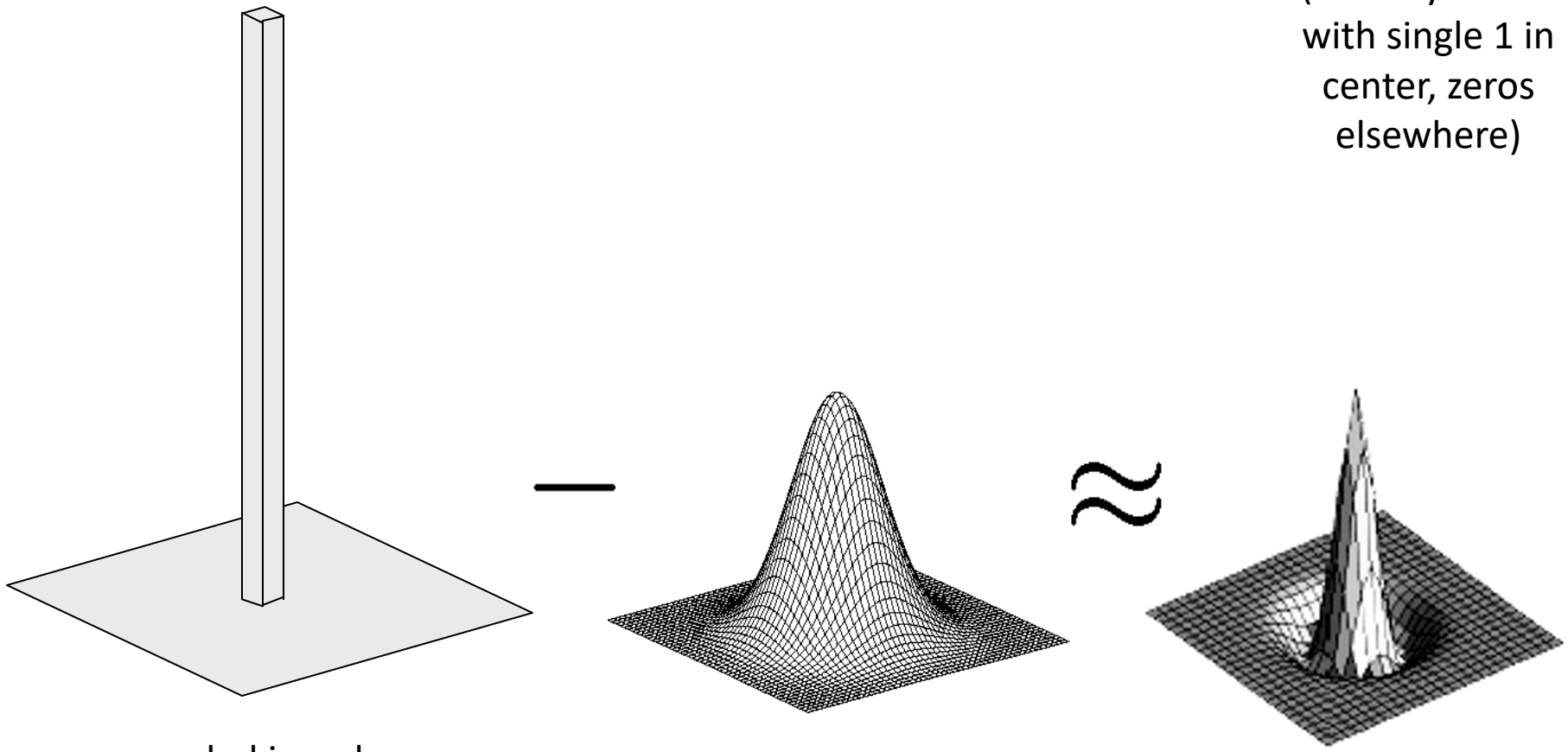
Sharpen filter

$$F + \alpha\,(F - \underbrace{F * H}_{\text{blurred image}}) = (1+\alpha)\,F - \alpha\,(F * H) = F * (\underbrace{[1+\alpha]\,e}_{\text{unit impulse}} - \alpha H)$$

image

blurred image

unit impulse (identity kernel with single 1 in center, zeros elsewhere)



scaled impulse

Gaussian

Sharpen filter

Edge Aware Smoothing: Gaussian vs Bilateral

Smooths everything nearby
(even edges)

Smooths everything nearby
(even edges)

$$h[m,n] = \sum_{k,l} g[k,l] f[m+k, n+l]$$

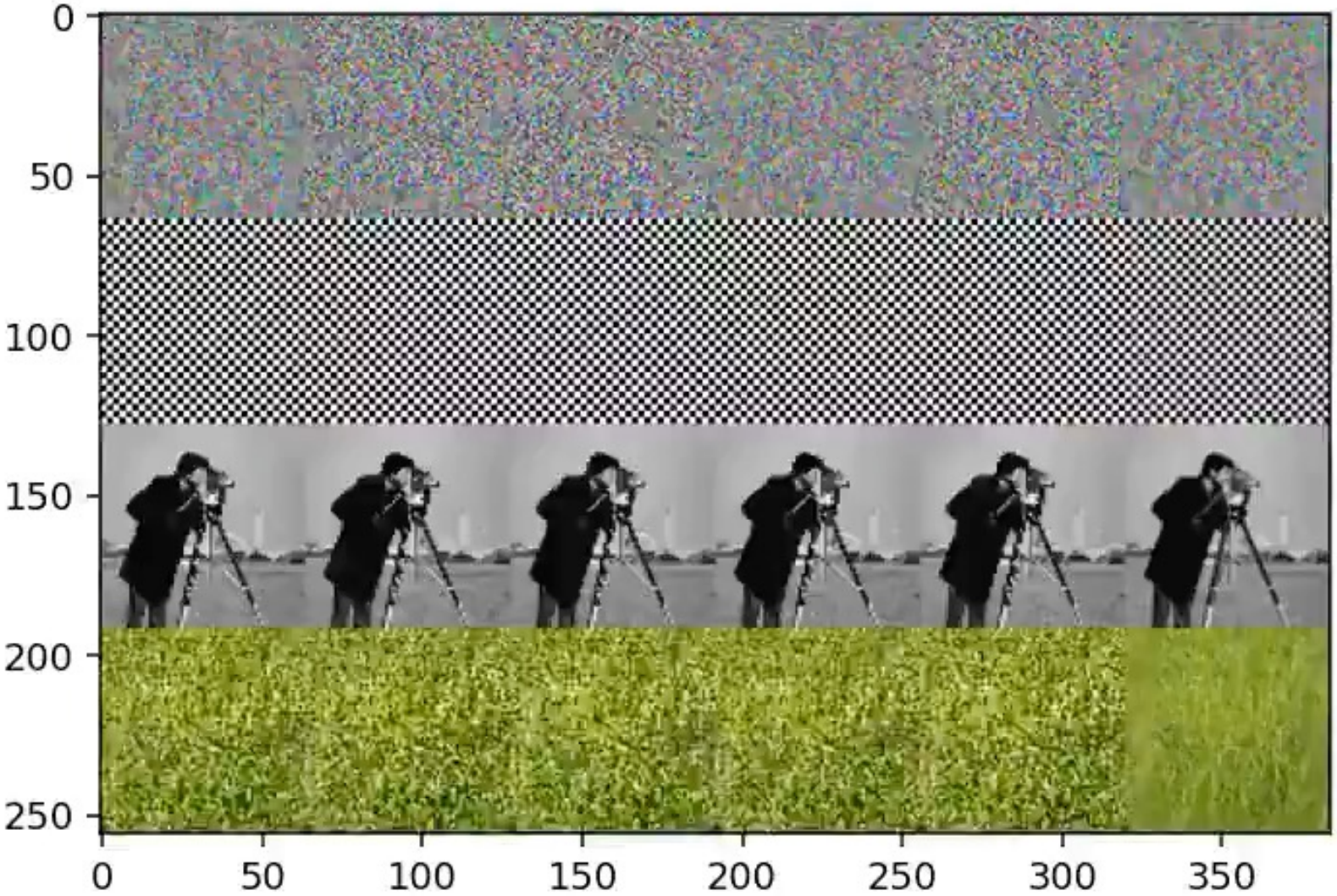
$$h[m,n] = \frac{1}{W_{mn}} \sum_{k,l} g[k,l] r_{mn}[k,l] f[m+k, n+l]$$

Aliasing

- Images are Signals in 2D
- Signals contain low frequency (smooth regions) and high frequency (sharp changes in intensity)
- To accurately downsample a signal/image, # of samples >= highest frequency in the signal. (Nyquist Rate!)
- If your task is to downsample by 1/4, you do not have enough samples, thus the downsampled image is inaccurate especially in terms of high frequency components.

A real problem!

128 x 128 → 64x64



Open-CV:
default, bicubic, Lanczos4

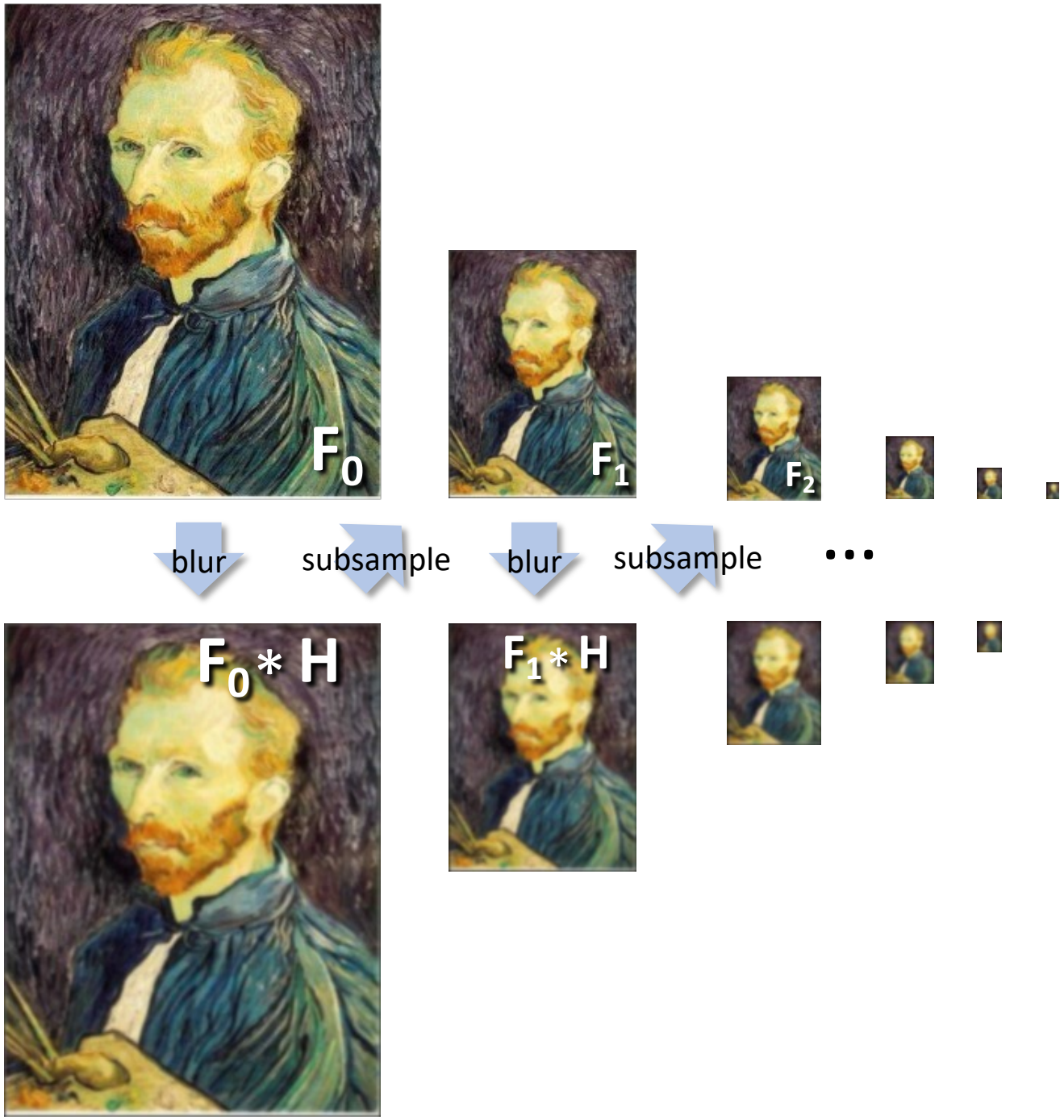
Pytorch:
bilinear, bicubic

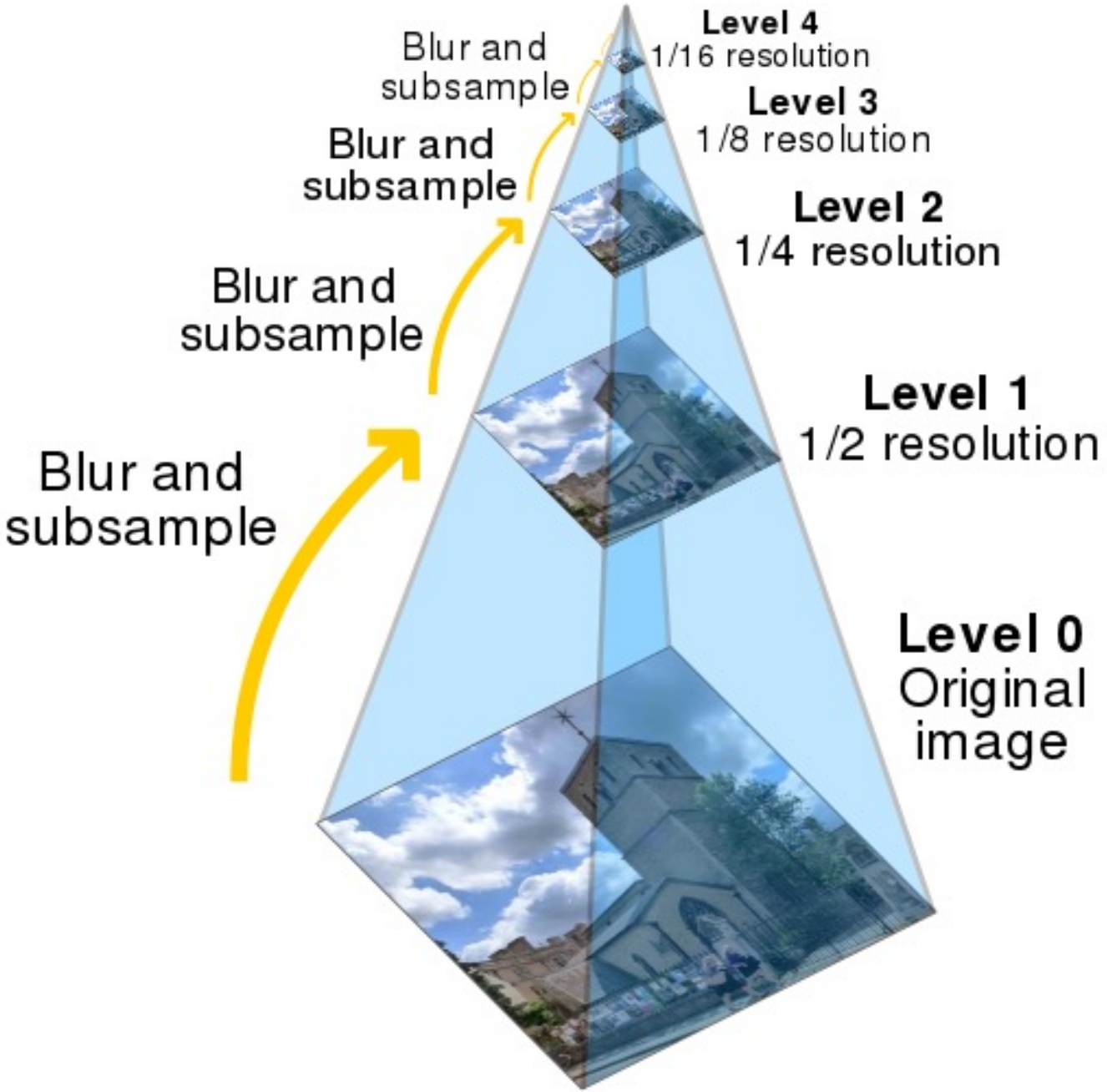
PIL: Lanczos

Credit: @jaakkolehtinen

Solution = Gaussian pre-filtering

- Solution: filter the image, *then* subsample





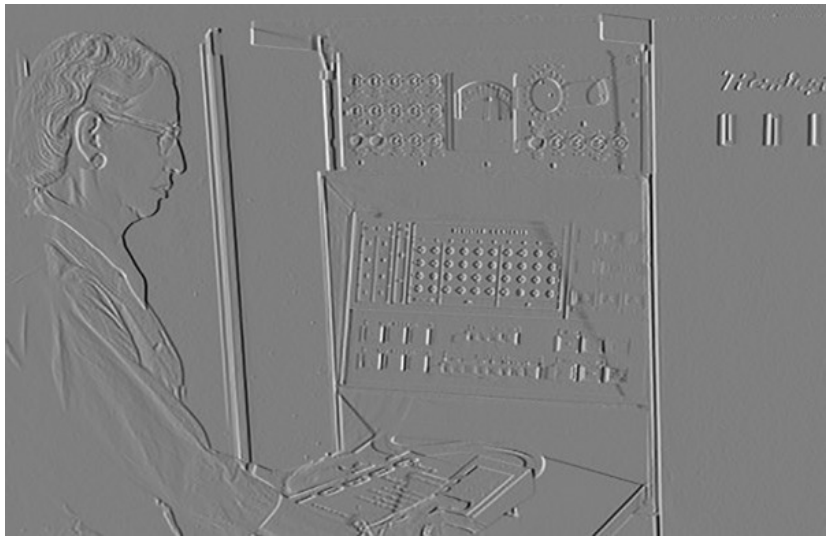
Gaussian Pyramids:

- Efficient representation for searching and sorting through a large (millions) volume of images.
- Very useful in image retrieval
- A powerful generic concept of representing images with hierarchical features capturing high-level details to low-level structures.

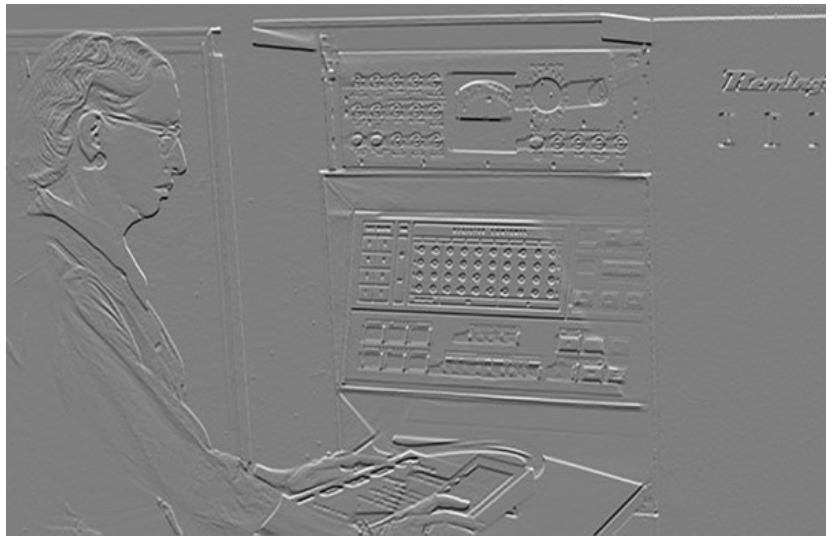
Partial Derivatives

Can be implemented as a convolution operation

-1	1



$$\frac{\partial f(x,y)}{\partial x}$$



$$\frac{\partial f(x,y)}{\partial y}$$

-1
1

or

1
-1

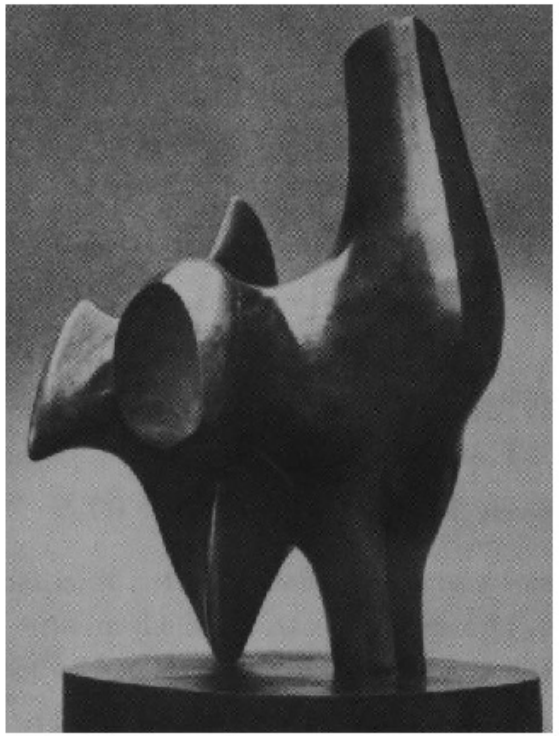
Today’s Lecture

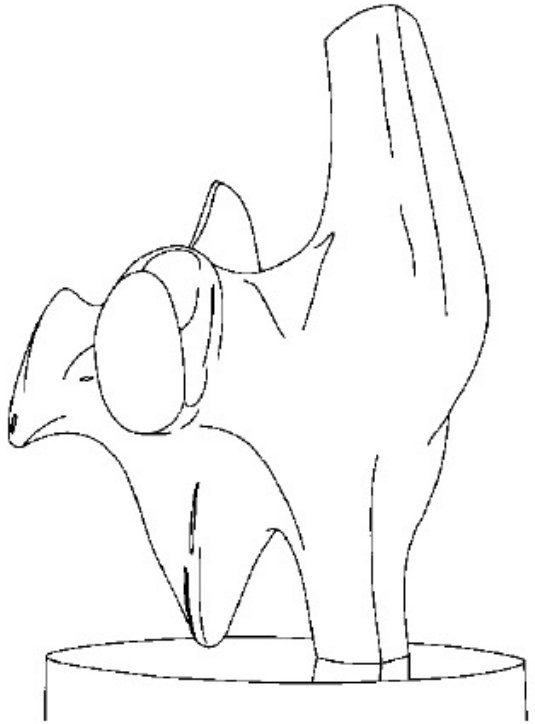
- Edge Detection
- Fourier Analysis (in 1D)
- Fourier Analysis (in 2D)
- Applications of Fourier Analysis
  - Hybrid Image
  - JPG Compression

Today's Lecture

- Edge Detection
- Fourier Analysis (in 1D)
- Fourier Analysis (in 2D)
- Applications of Fourier Analysis
  - Hybrid Image
  - JPG Compression

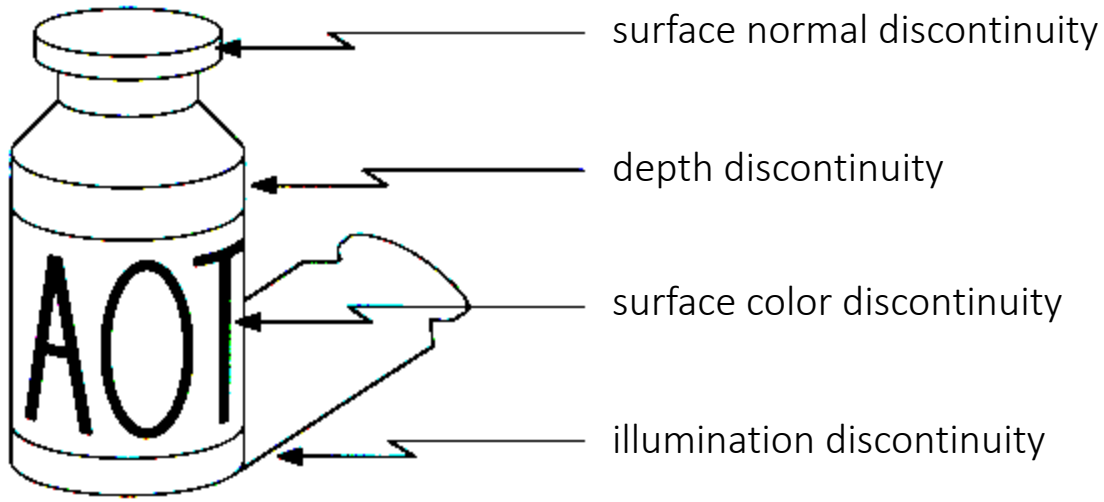
Edge detection





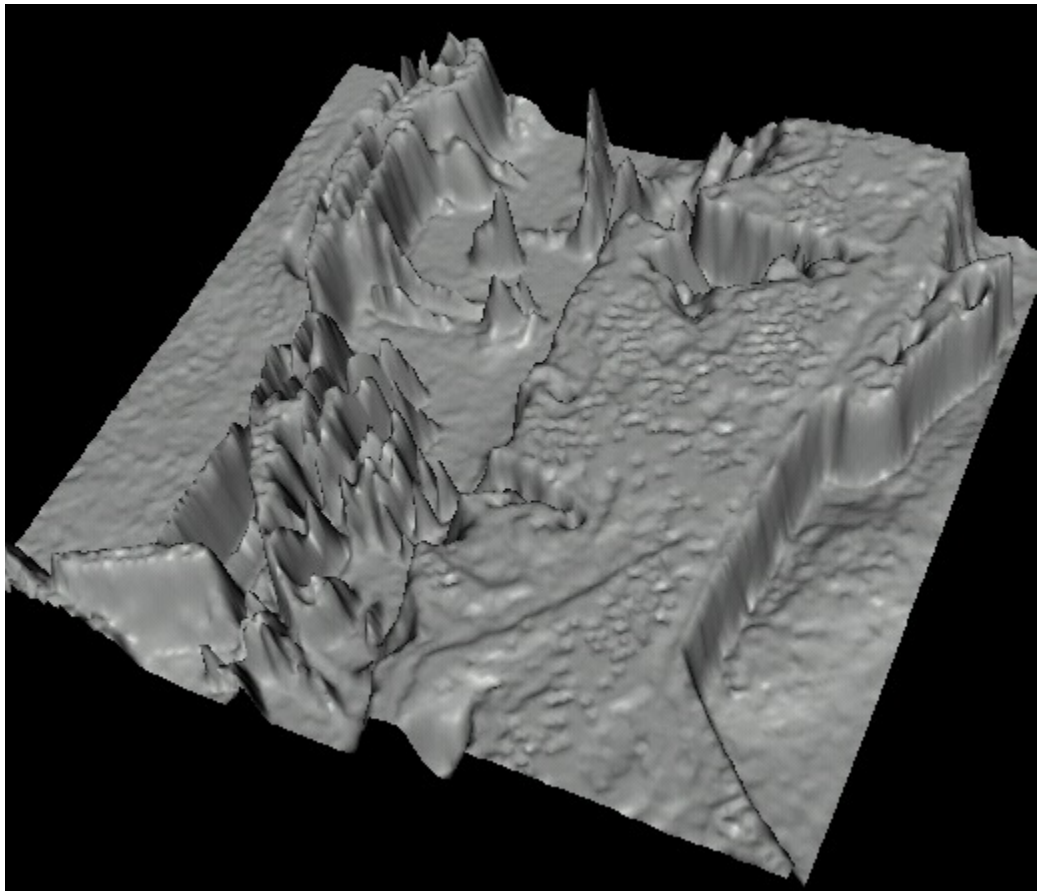
- Convert a 2D image into a set of curves
  - Extracts salient features of the scene
  - More compact than pixels

Origin of edges



- Edges are caused by a variety of factors

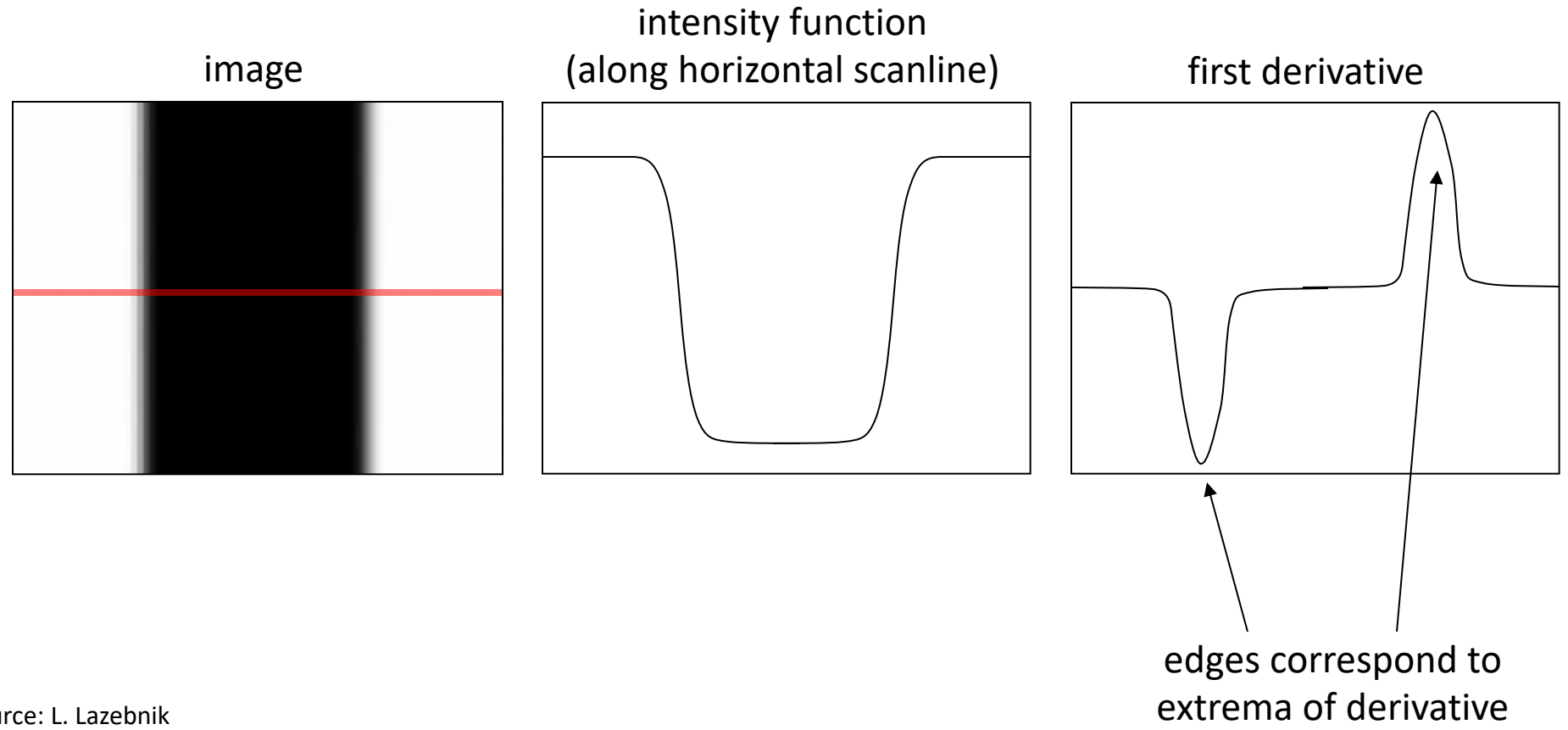
Images as functions…



- Edges look like steep cliffs

Characterizing edges

- An edge is a place of *rapid change* in the image intensity function



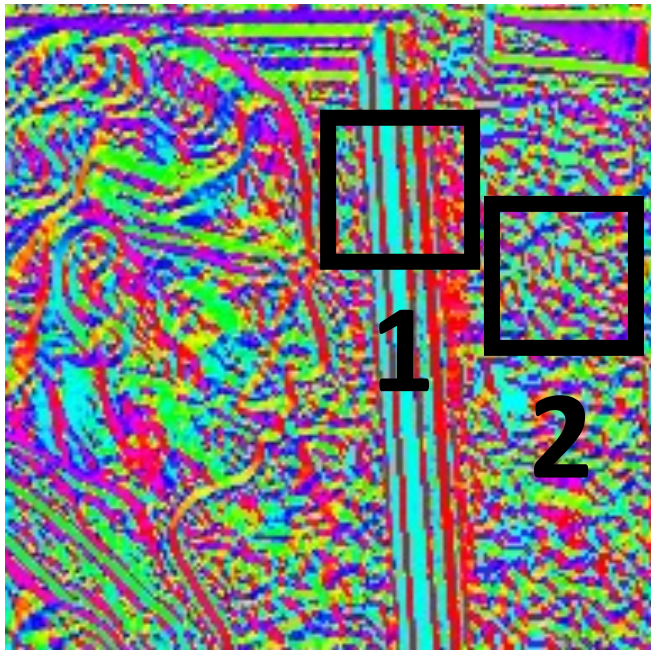
Source: L. Lazebnik

Image Gradient & Edges

$$\theta = \tan^{-1}\left(\frac{\partial f}{\partial y} / \frac{\partial f}{\partial x}\right)$$

Direction of image gradients

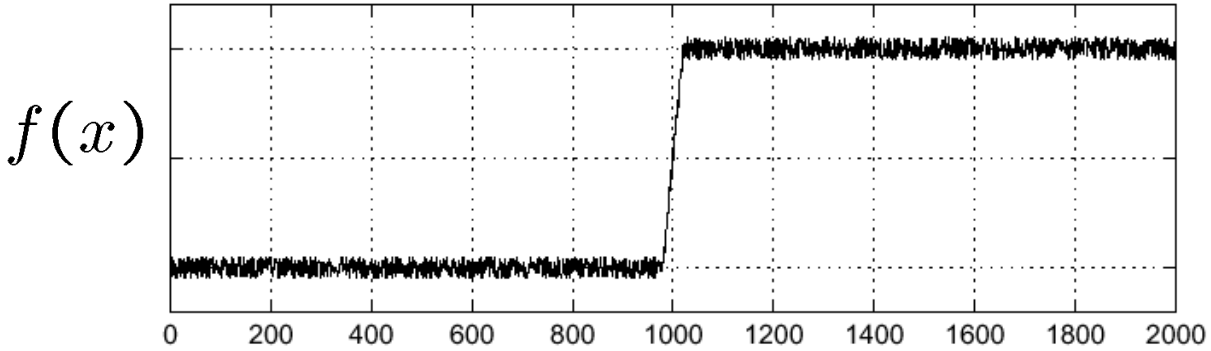
Why is there structure at 1 and not at 2?

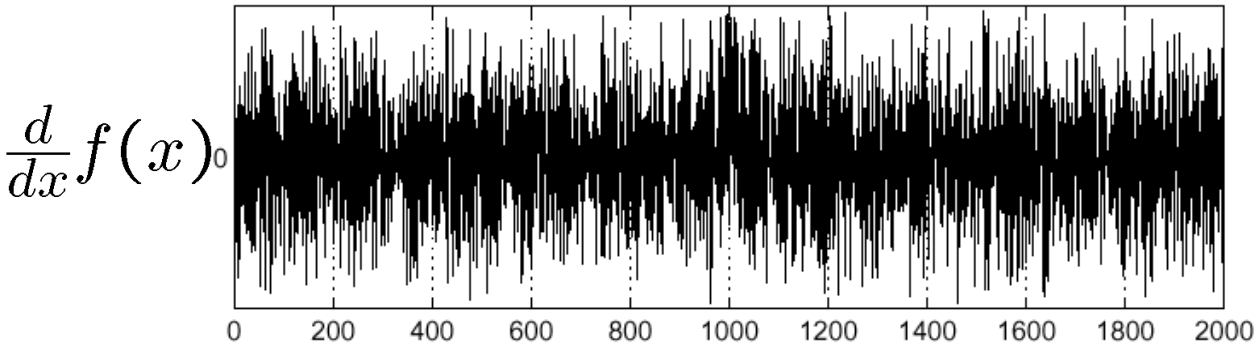


Source: D. Fouhey

Effects of noise

- Consider a single row or column of the image
  - Plotting intensity as a function of position gives a signal

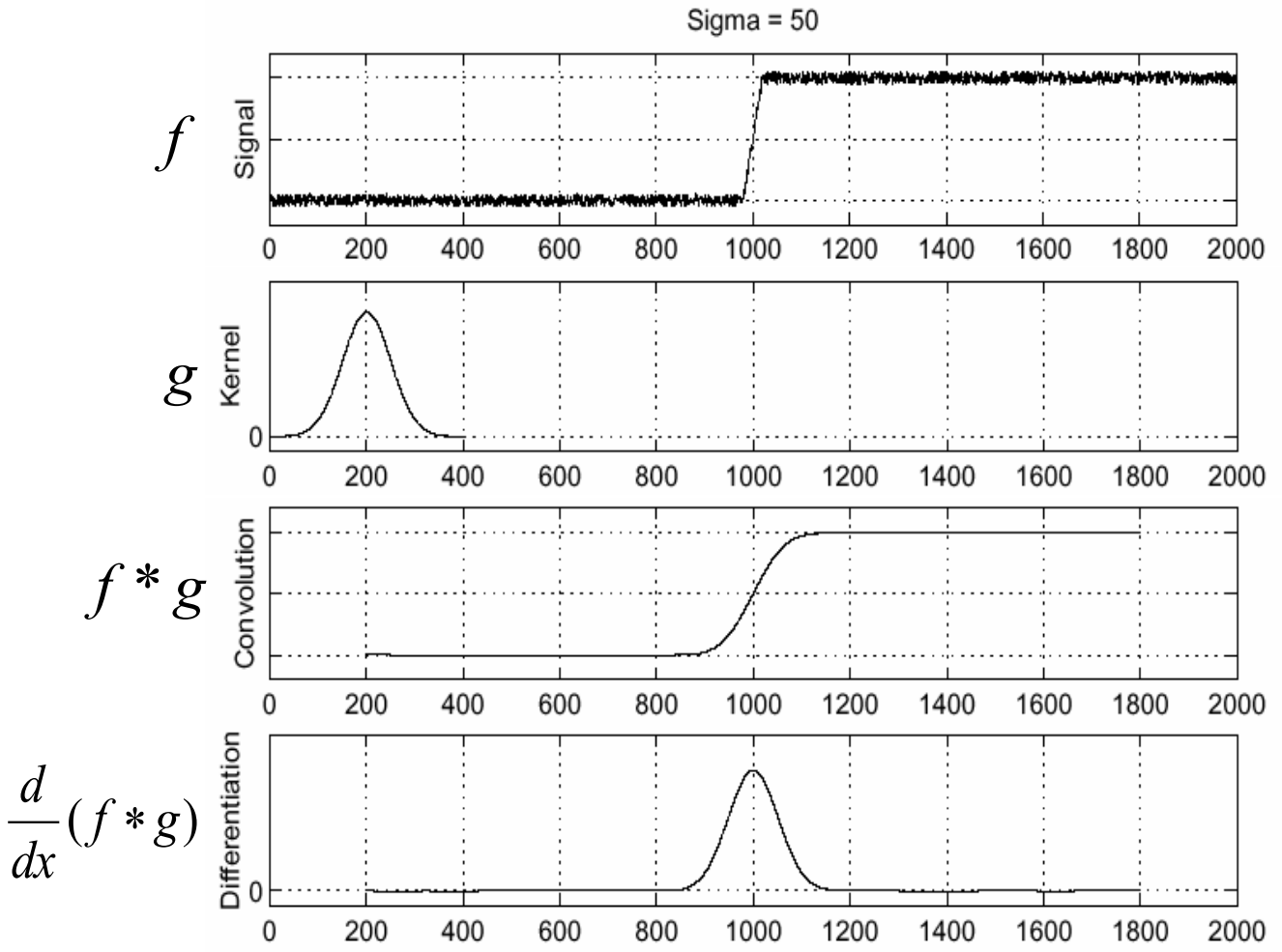




Where is the edge?

Source: S. Seitz

Solution: smooth first



- To find edges, look for peaks in $\frac{d}{dx}(f * g)$

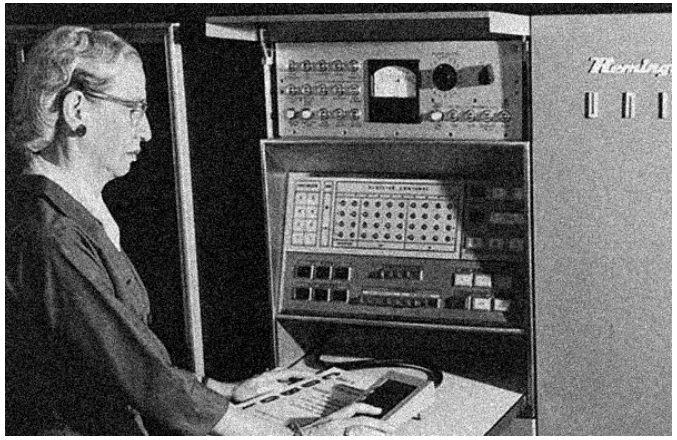
Source: S. Seitz

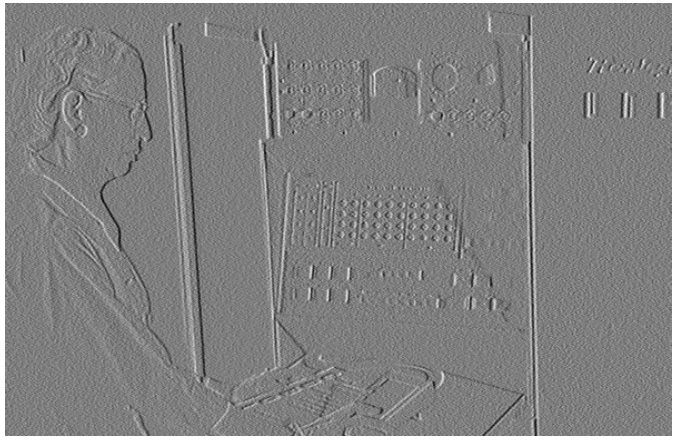
Noise in 2D

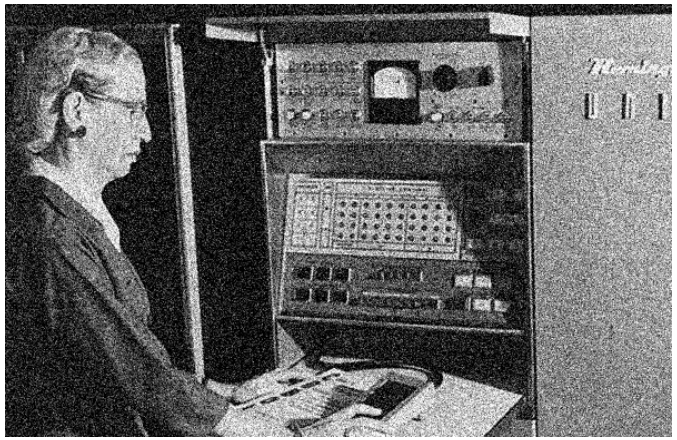
Noisy Input

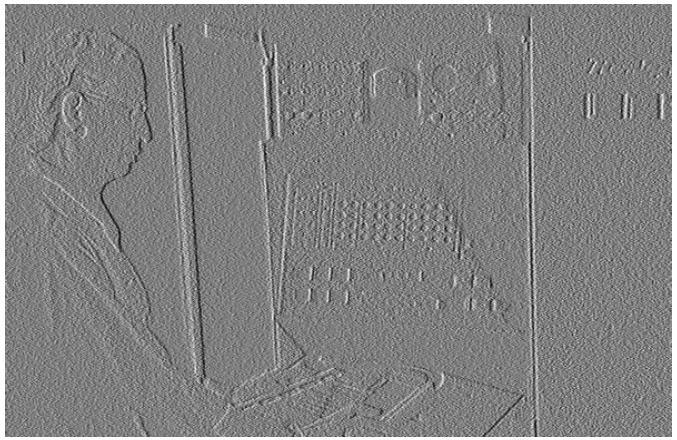
Ix via [-1,0,1]

Zoom









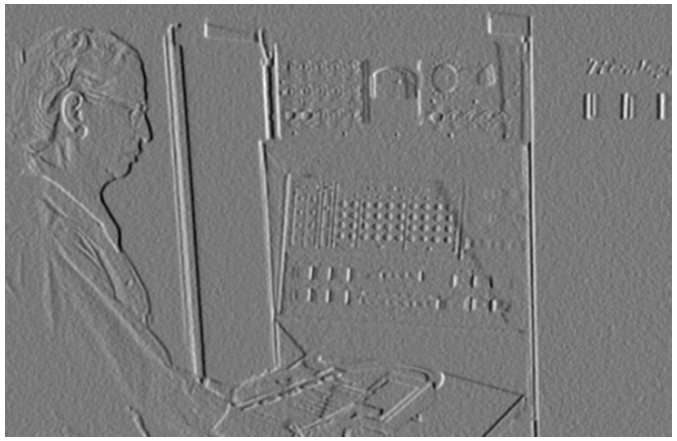
Source: D. Fouhey

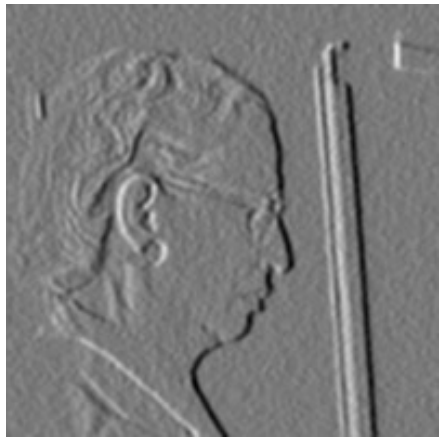
Noise + Smoothing

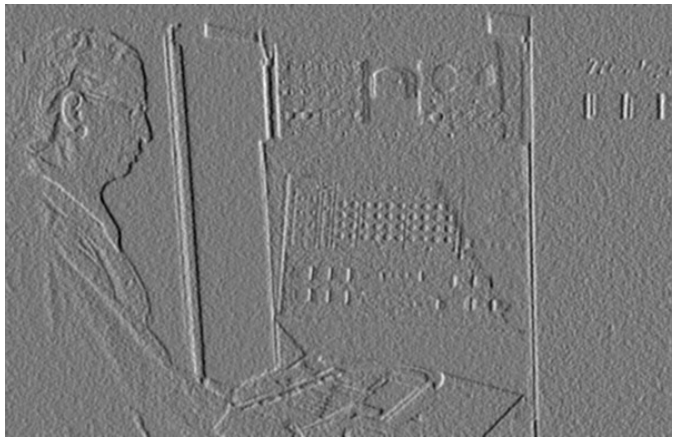
Smoothed Input

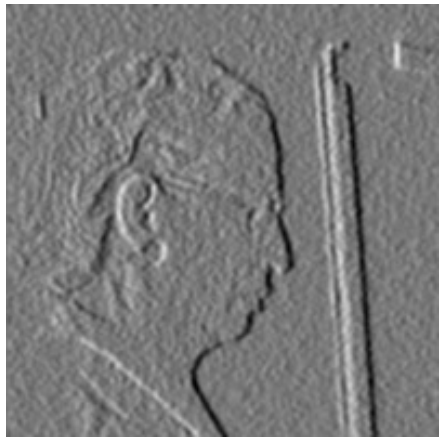
Ix via [-1,0,1]

Zoom



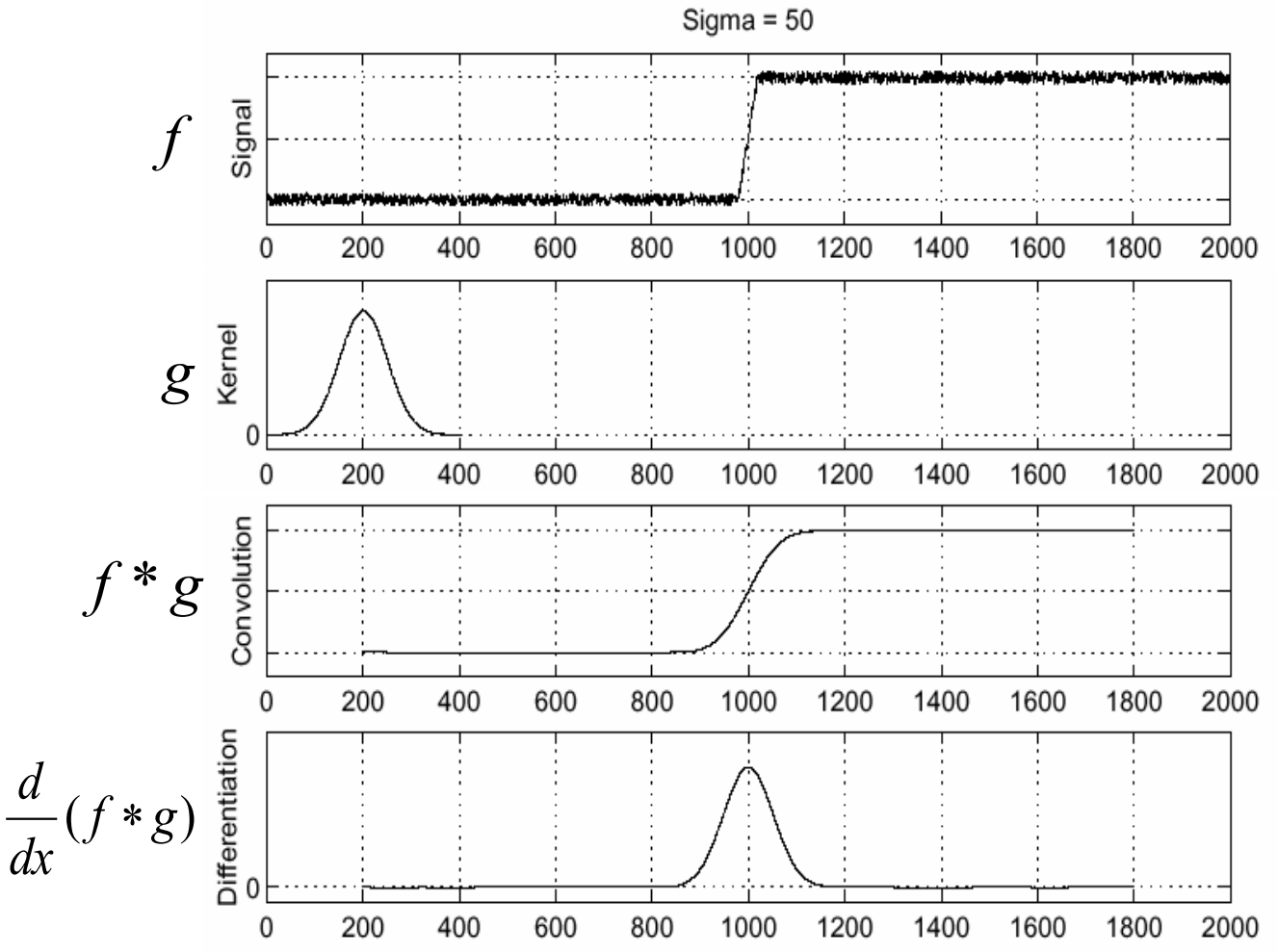






Source: D. Fouhey

How many convolutions here?



can we reduce this?

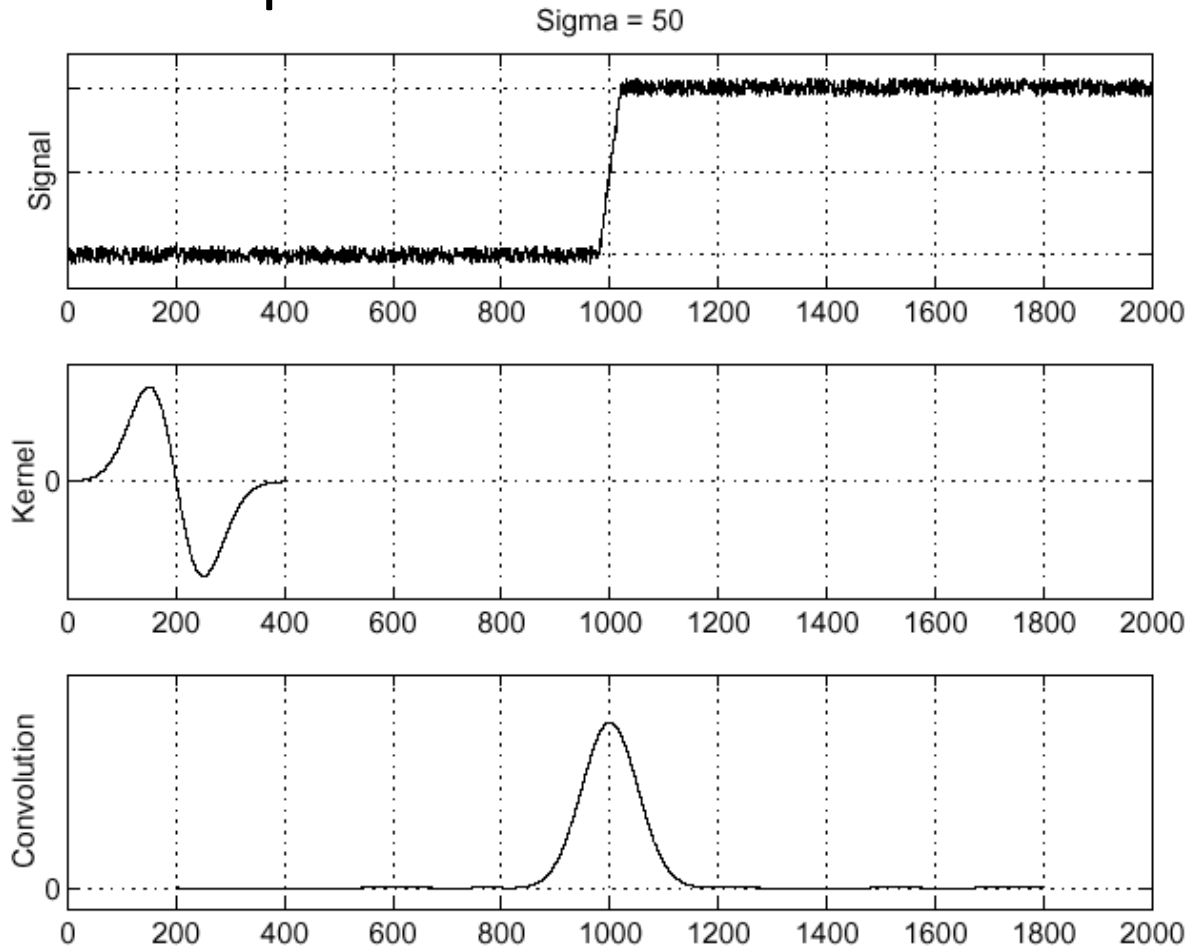
Derivative theorem of convolution

$$\frac{\partial}{\partial x}(h \star f) = (\frac{\partial}{\partial x}h) \star f$$

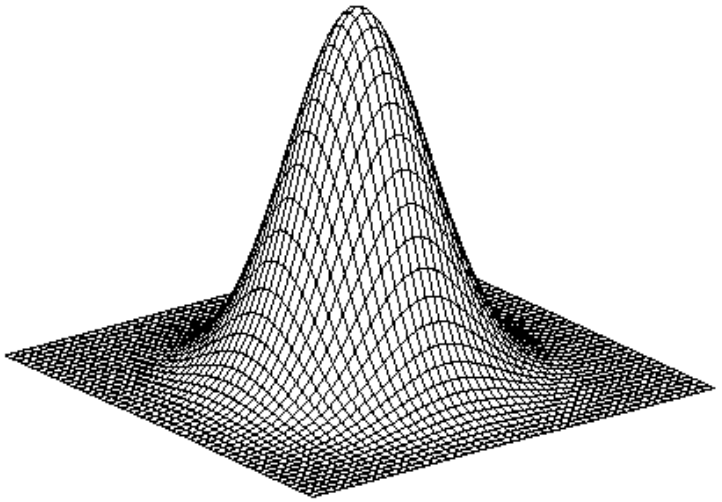
- This saves us one operation:

$\frac{\partial}{\partial x}h$

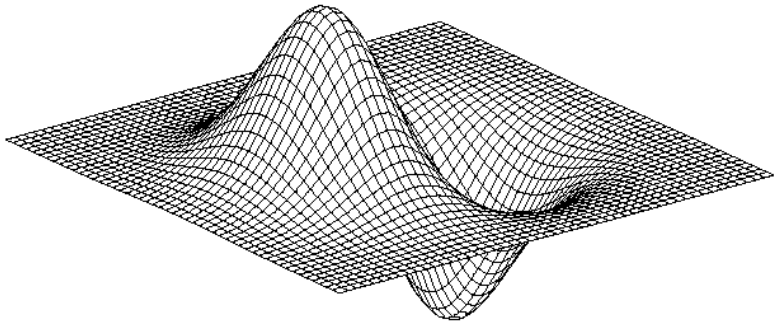
$(\frac{\partial}{\partial x}h) \star f$



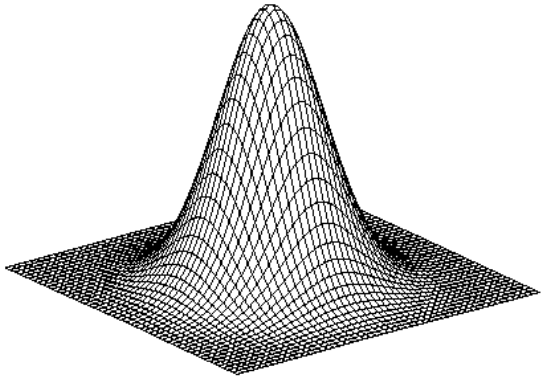
Derivative of Gaussian filter



* [1 -1] =

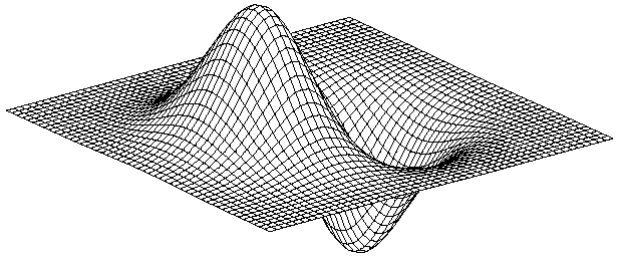


2D edge detection filters



Gaussian

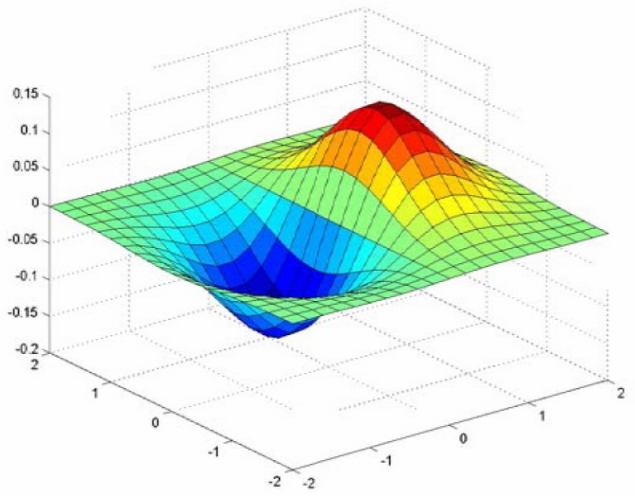
$$h_\sigma(u,v) = \frac{1}{2\pi\sigma^2} e^{-\frac{u^2+v^2}{2\sigma^2}}$$



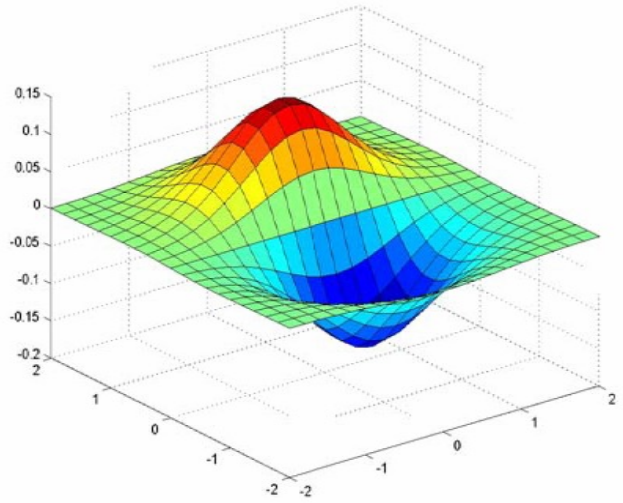
derivative of Gaussian (x)

$$\frac{\partial}{\partial x} h_\sigma(u,v)$$

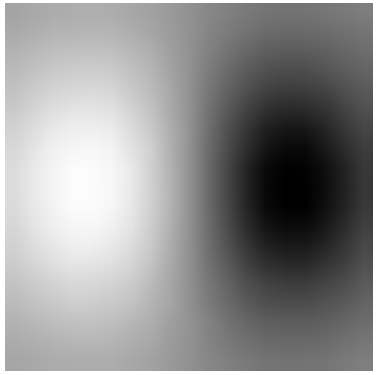
Derivative of Gaussian filter

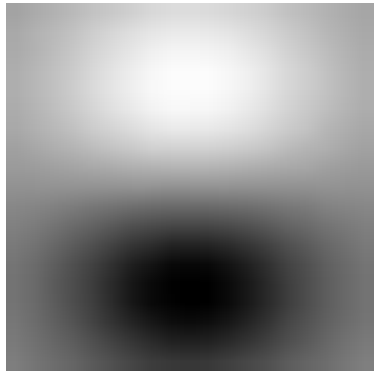


x-direction



y-direction





The Sobel operator

- Common approximation of derivative of Gaussian

$\frac{1}{8}$

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

s_x

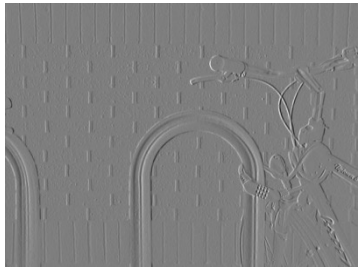
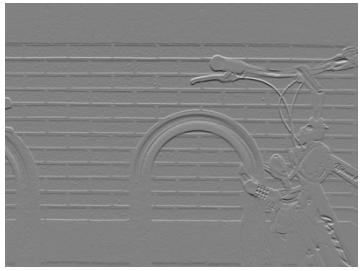
$\frac{1}{8}$

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

s_y

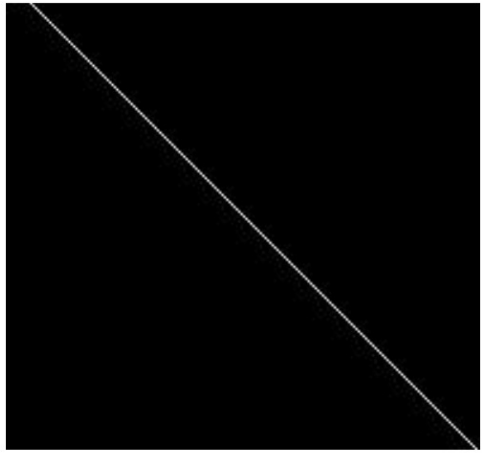
- The standard definition of the Sobel operator omits the 1/8 term
  - doesn't make a difference for edge detection
  - the 1/8 term **is** needed to get the right gradient magnitude

Sobel operator: example

Source: Wikipedia

Image with Edge



Edge Location

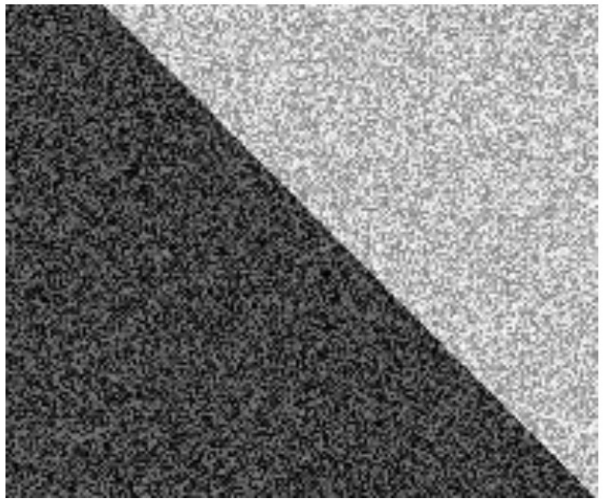
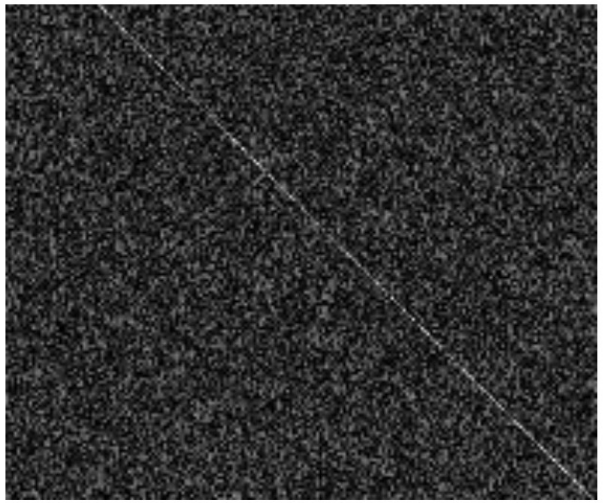
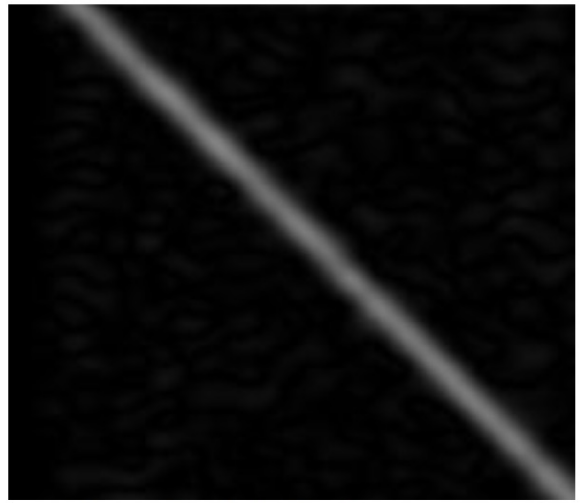


Image + Noise



Derivatives detect
edge *and* noise



Smoothed derivative removes
noise, but blurs edge

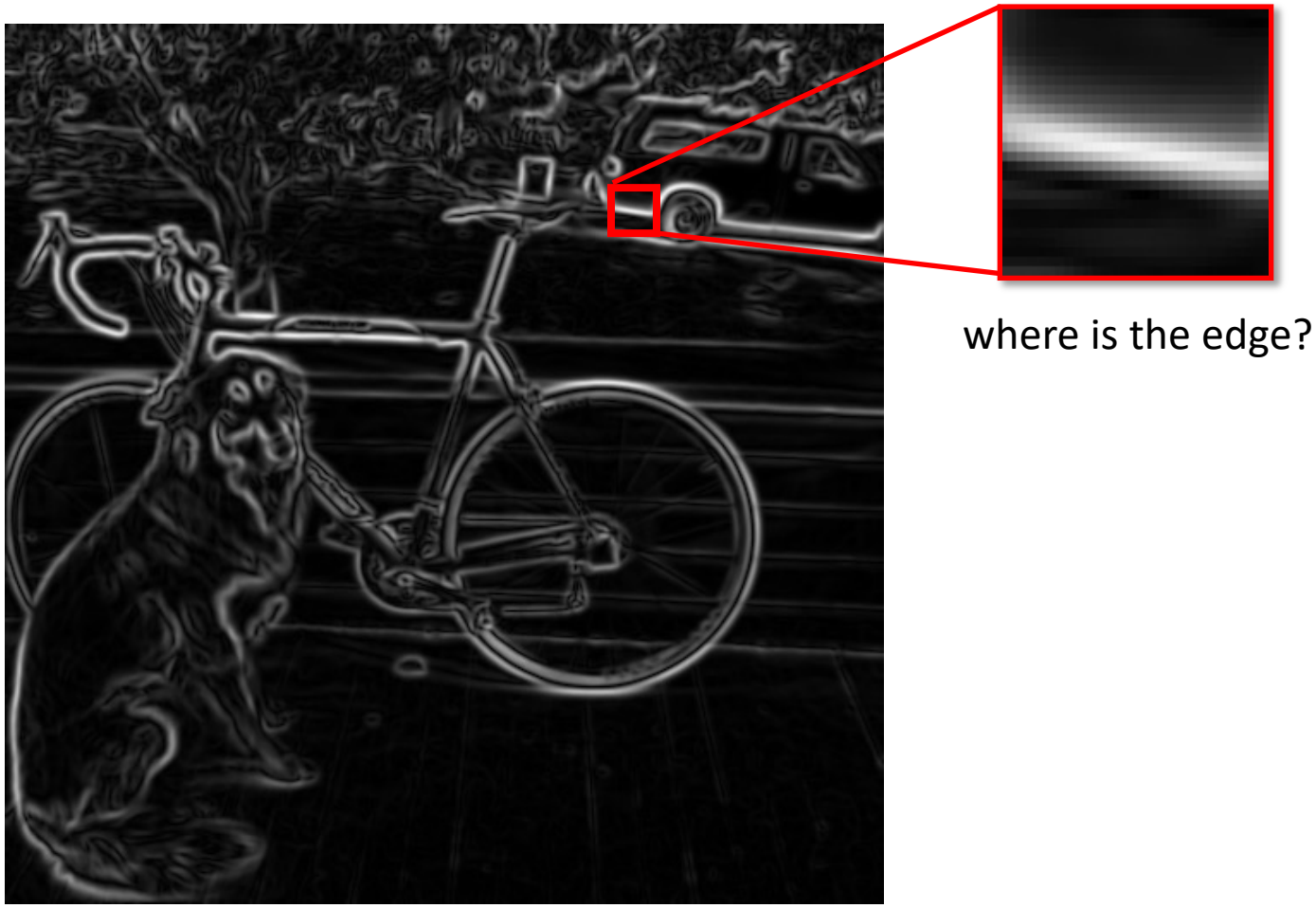
Example

original image

Demo: http://bigwww.epfl.ch/demo/ip/demos/edgeDetector/

Image credit: Joseph Redmon

Finding edges

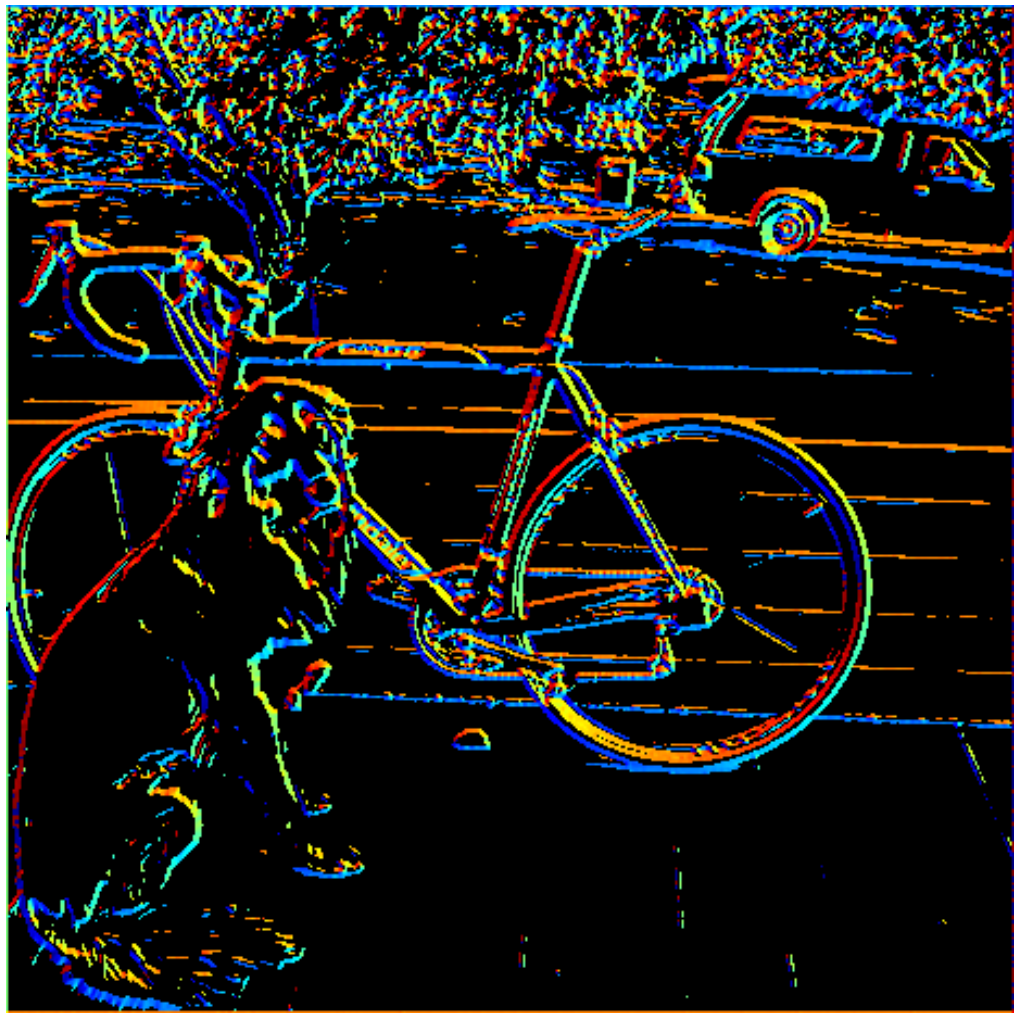


where is the edge?

smoothed gradient magnitude

Get Orientation at Each Pixel

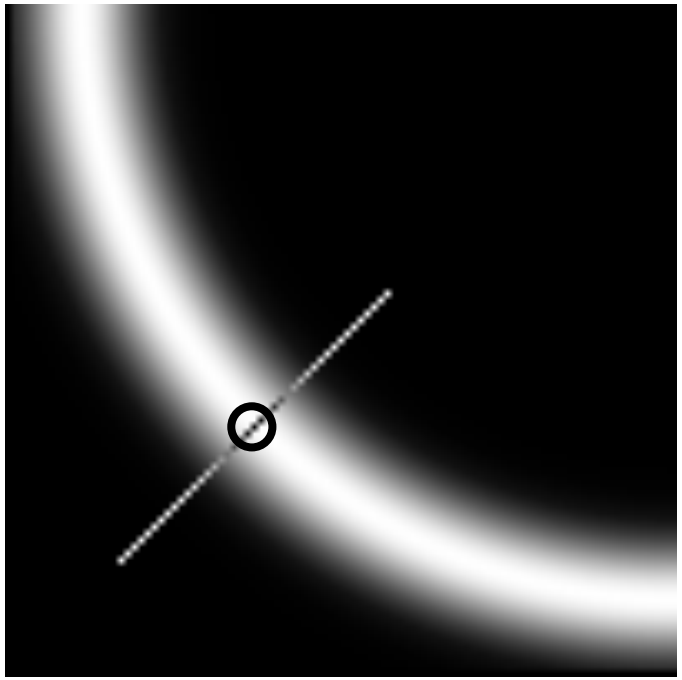
- Get orientation (below, threshold at minimum gradient magnitude)

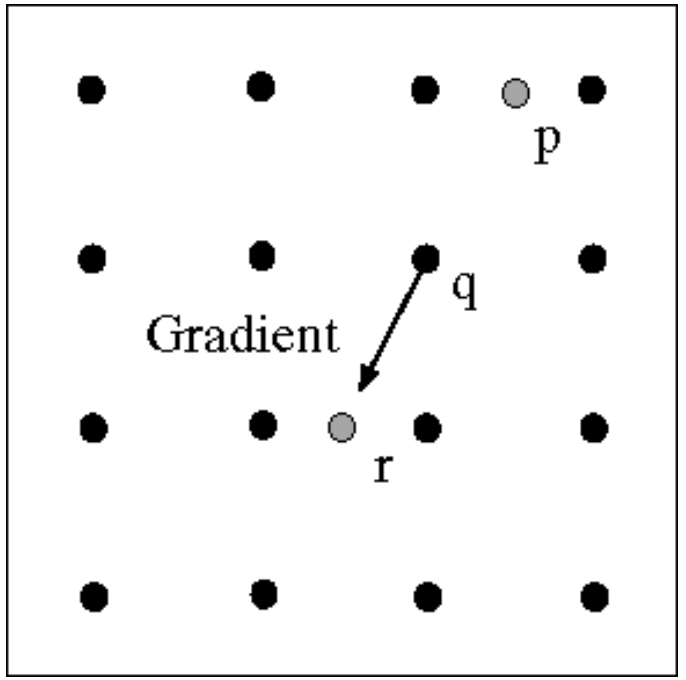


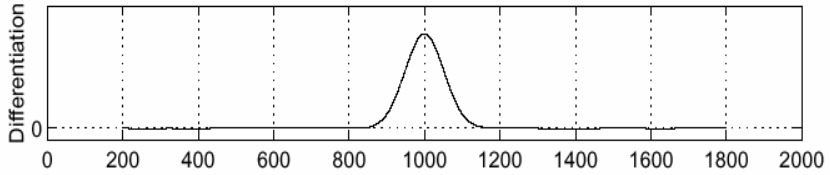
theta = atan2(gy, gx)

Gradient orientation angle

Non-maximum supression





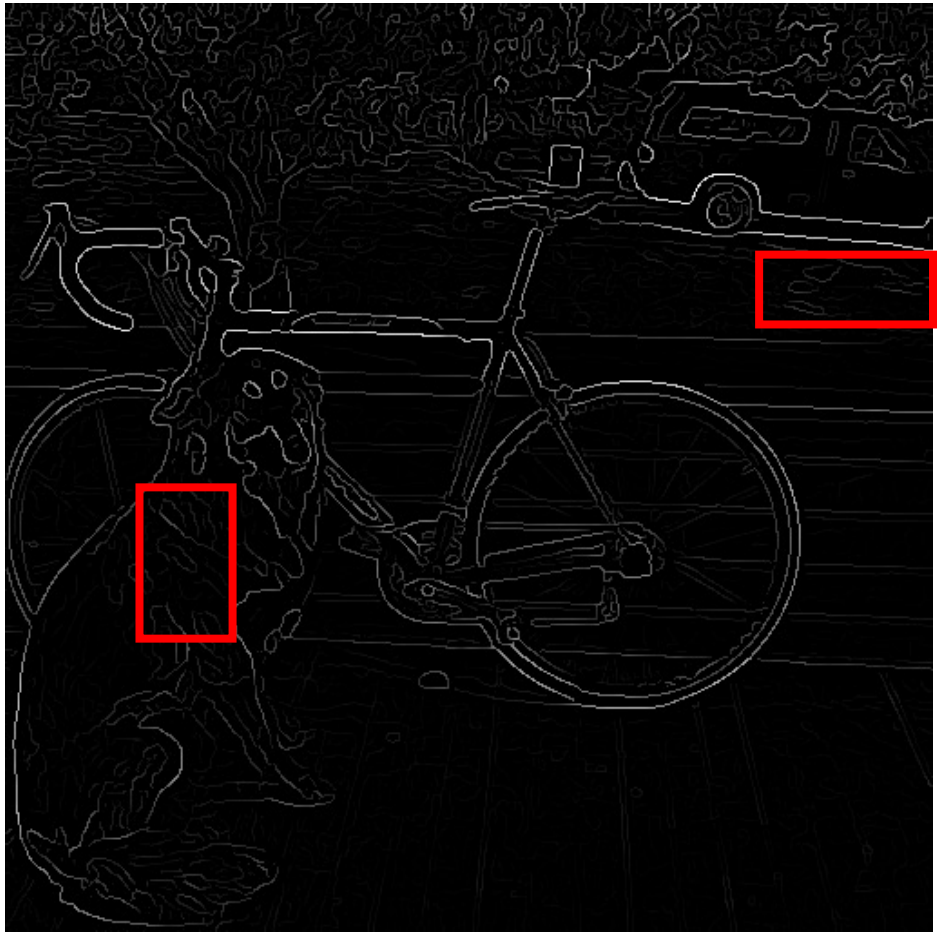


- Check if pixel is local maximum along gradient direction
  - requires *interpolating* pixels p and r

Before Non-max Suppression

After Non-max Suppression

Still noise exists!



Thresholding edges

- Still some noise
- Only want strong edges
- 2 thresholds, 3 cases
  - R > T: strong edge
  - R < T but R > t: weak edge
  - R < t: no edge

- Strong edges are edges!
- Weak edges are edges
  iff they connect to strong
  - Look in some neighborhood
    (usually 8 closest)

Canny edge detector

1. Filter image with derivative of Gaussian
2. Find magnitude and orientation of gradient
3. Non-maximum suppression
4. Linking and thresholding (hysteresis):
   - Define two thresholds: low and high
   - Use the high threshold to start edge curves and the low threshold to continue them

Source: D. Lowe, L. Fei-Fei, J. Redmon

Canny edge detector

- Our first computer vision pipeline!
- Still a widely used edge detector in computer vision

J. Canny, ***A Computational Approach To Edge Detection***, IEEE Trans. Pattern Analysis and Machine Intelligence, 8:679-714, 1986.

- Depends on several parameters:

high threshold
low threshold

σ : width of the Gaussian blur

Canny edge detector

original

Canny with $\sigma = 1$

Canny with $\sigma = 2$

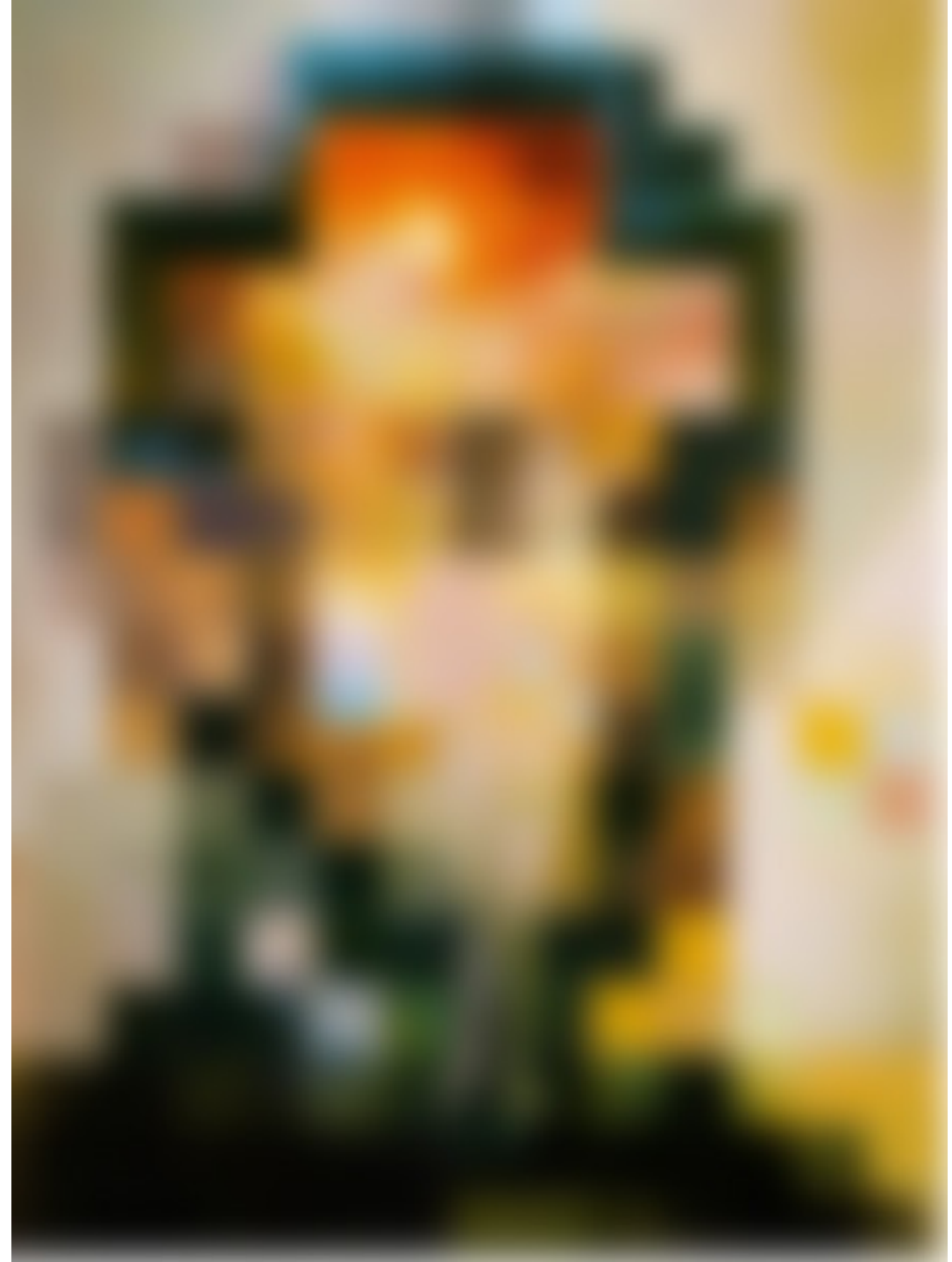
- The choice of σ depends on desired behavior
  - large σ detects "large-scale" edges
  - small σ detects fine edges

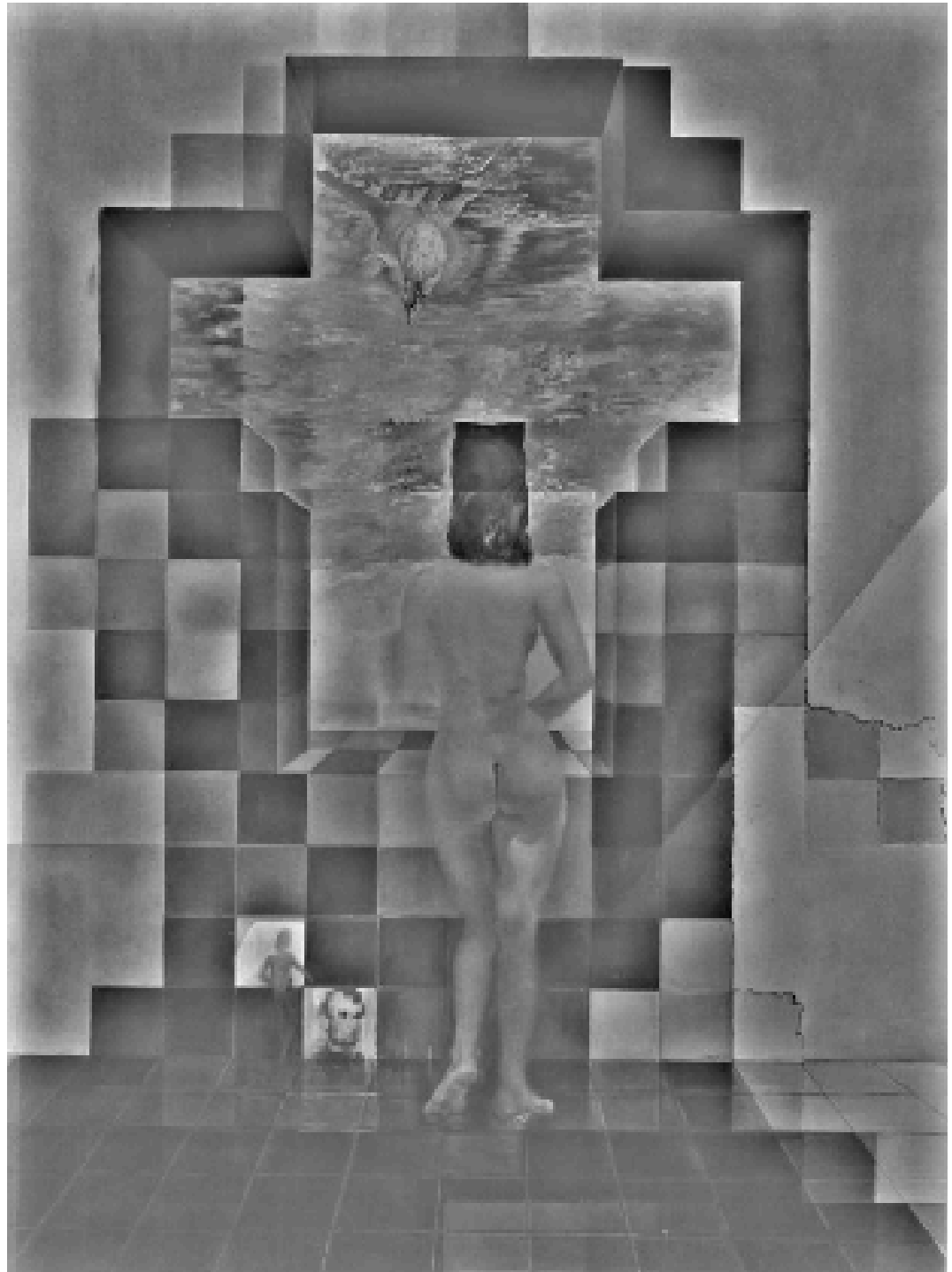
Source: S. Seitz

Today’s Lecture

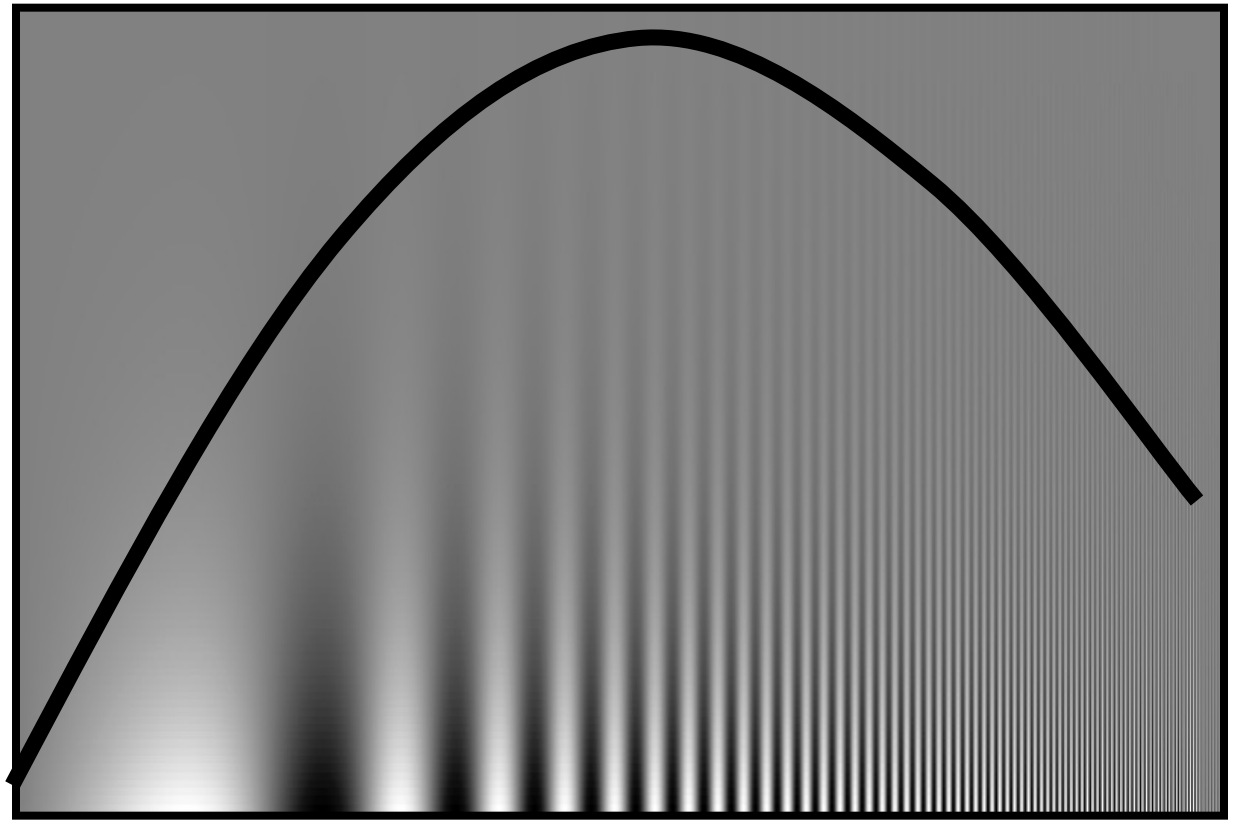
- Edge Detection
- Fourier Analysis (in 1D)
- Fourier Analysis (in 2D)
- Applications of Fourier Analysis
  - Hybrid Image
  - JPG Compression

Salvador Dali
“Gala Contemplating the Mediterranean Sea, which at 30 meters becomes the portrait of Abraham Lincoln”, 1976





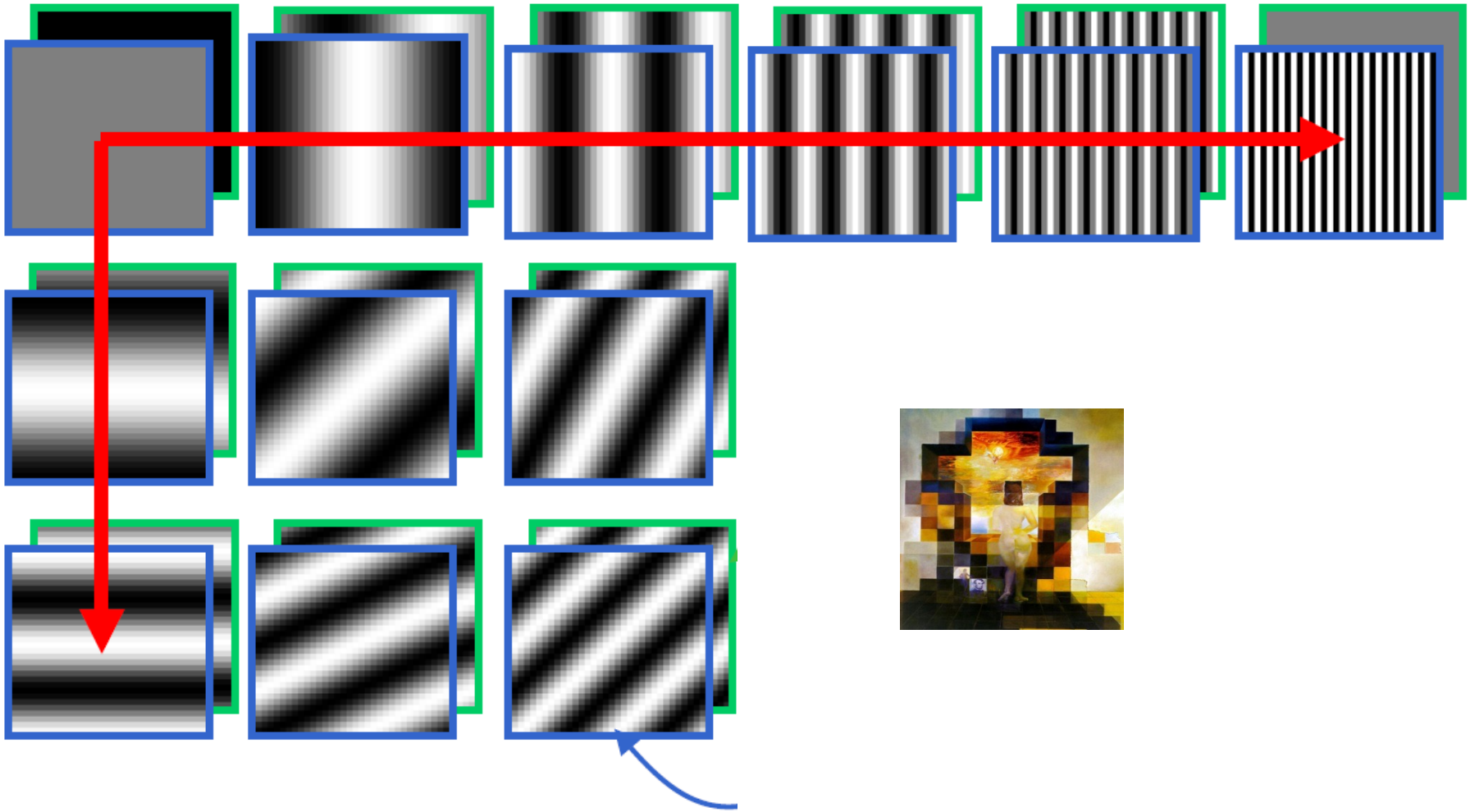
Spatial Frequencies and Perception



Campbell-Robson contrast sensitivity curve

A nice set of basis

Teases away fast vs. slow changes in the image.

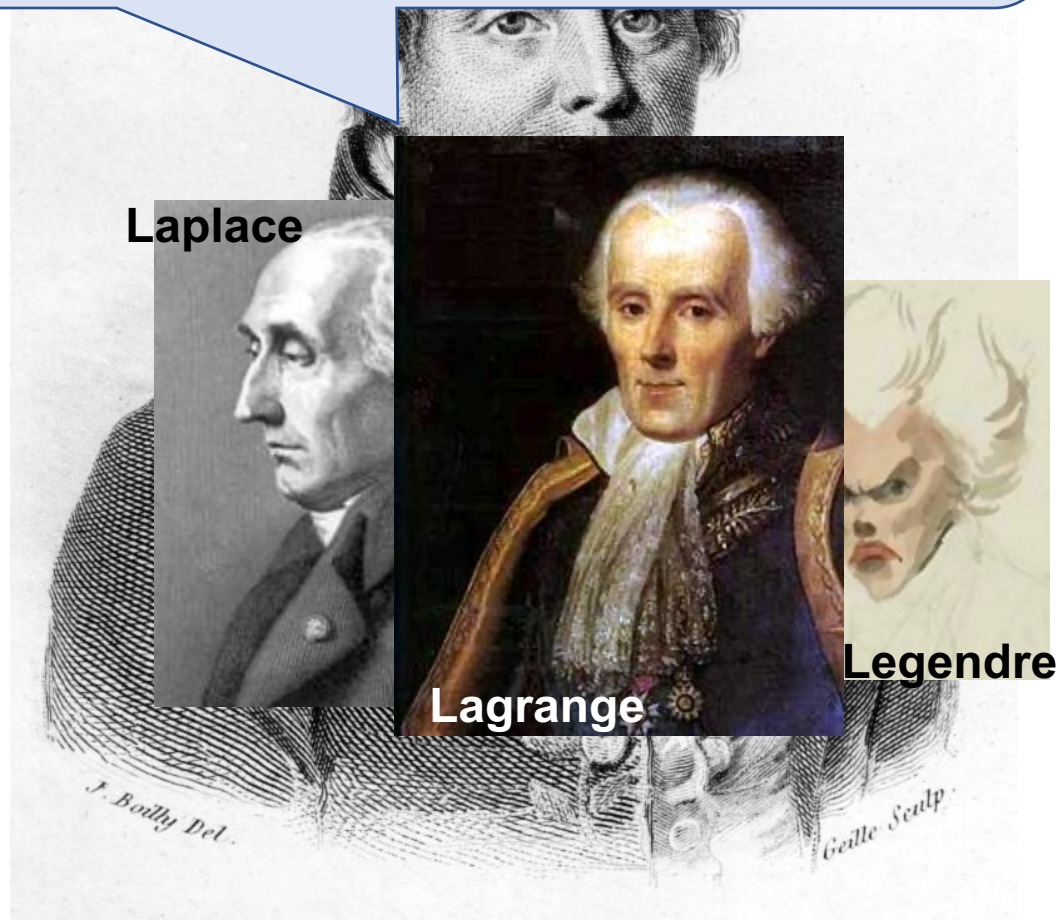


This change of basis has a special name…

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.*
- 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

...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.



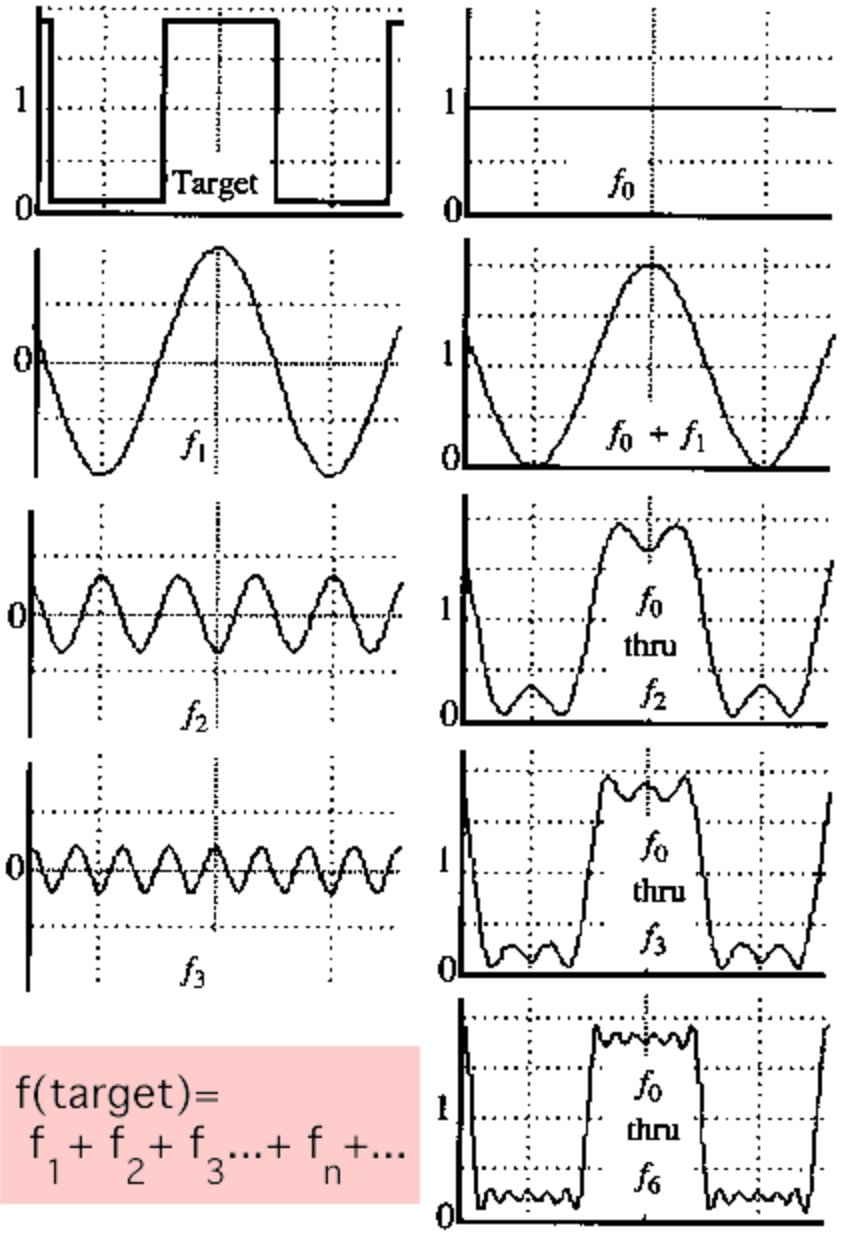
A sum of sines

- Our building block:
- $A\sin(\omega x + \phi)$

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

- How many degrees of freedom?

- What does each control?



Fourier Transform

•We want to understand the frequency ω of our signal. So, let's reparametrize the signal by ω instead of *x*:

For every ω from 0 to inf, ***F(ω)*** holds the amplitude *A* and phase ϕ of the corresponding sine $A\sin(\omega x + \phi)$

- How does *F* hold both?

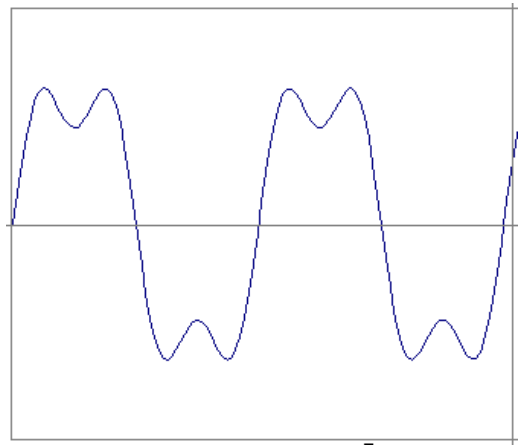
$$F(\omega) = R(\omega) + iI(\omega)$$

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

We can always go back:

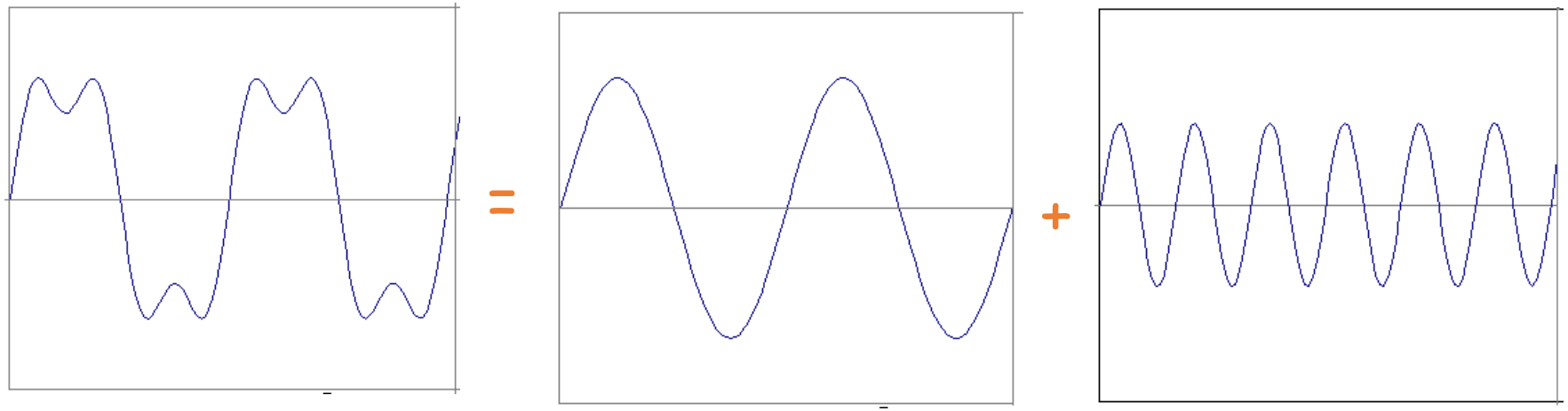
Time and Frequency

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



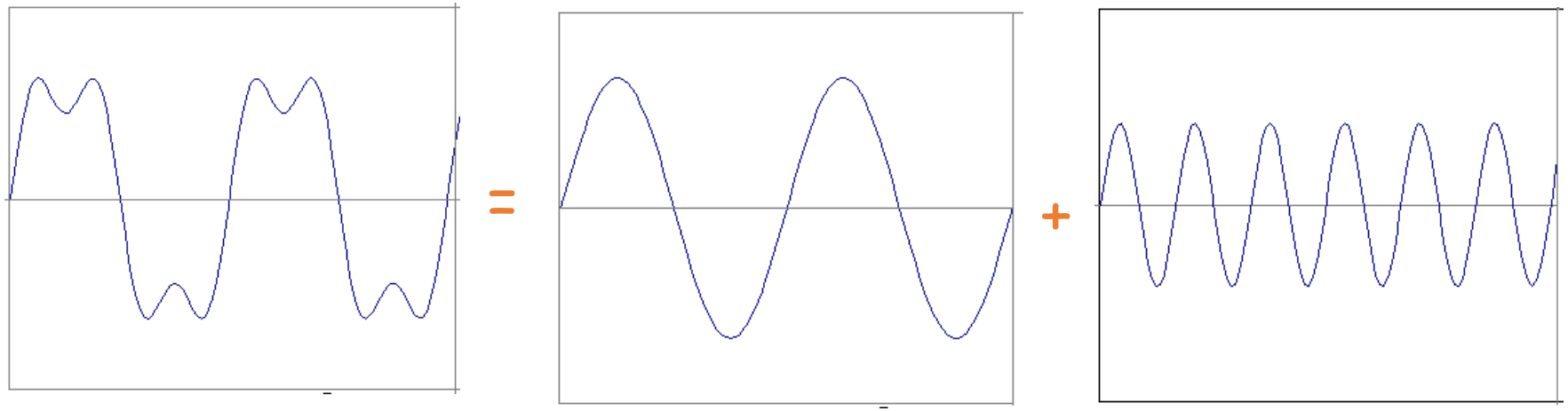
Time and Frequency

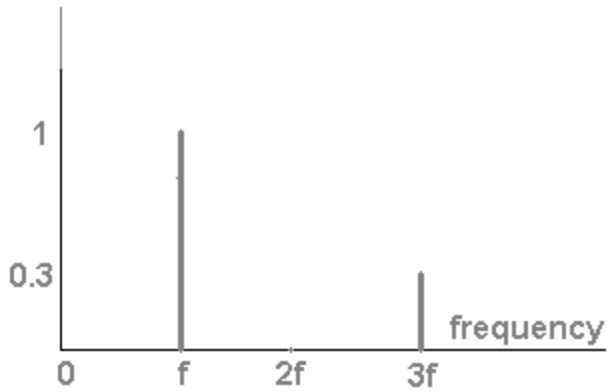
- example : $g(t) = \sin(2pf\,t) + (1/3)\sin(2p(3f)\,t)$



Frequency Spectra

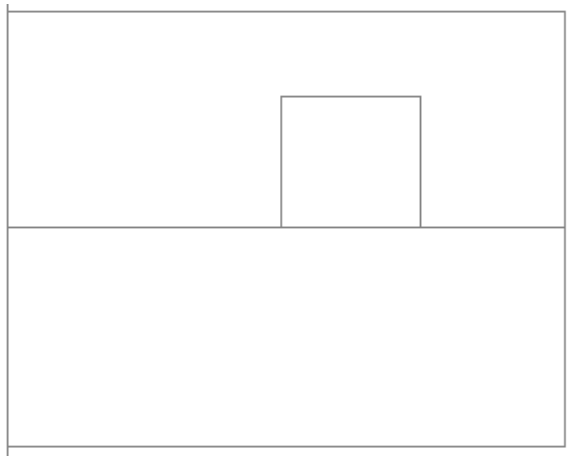
- example : $g(t) = \sin(2\pi f t) + (1/3)\sin(2\pi(3f)\, t)$



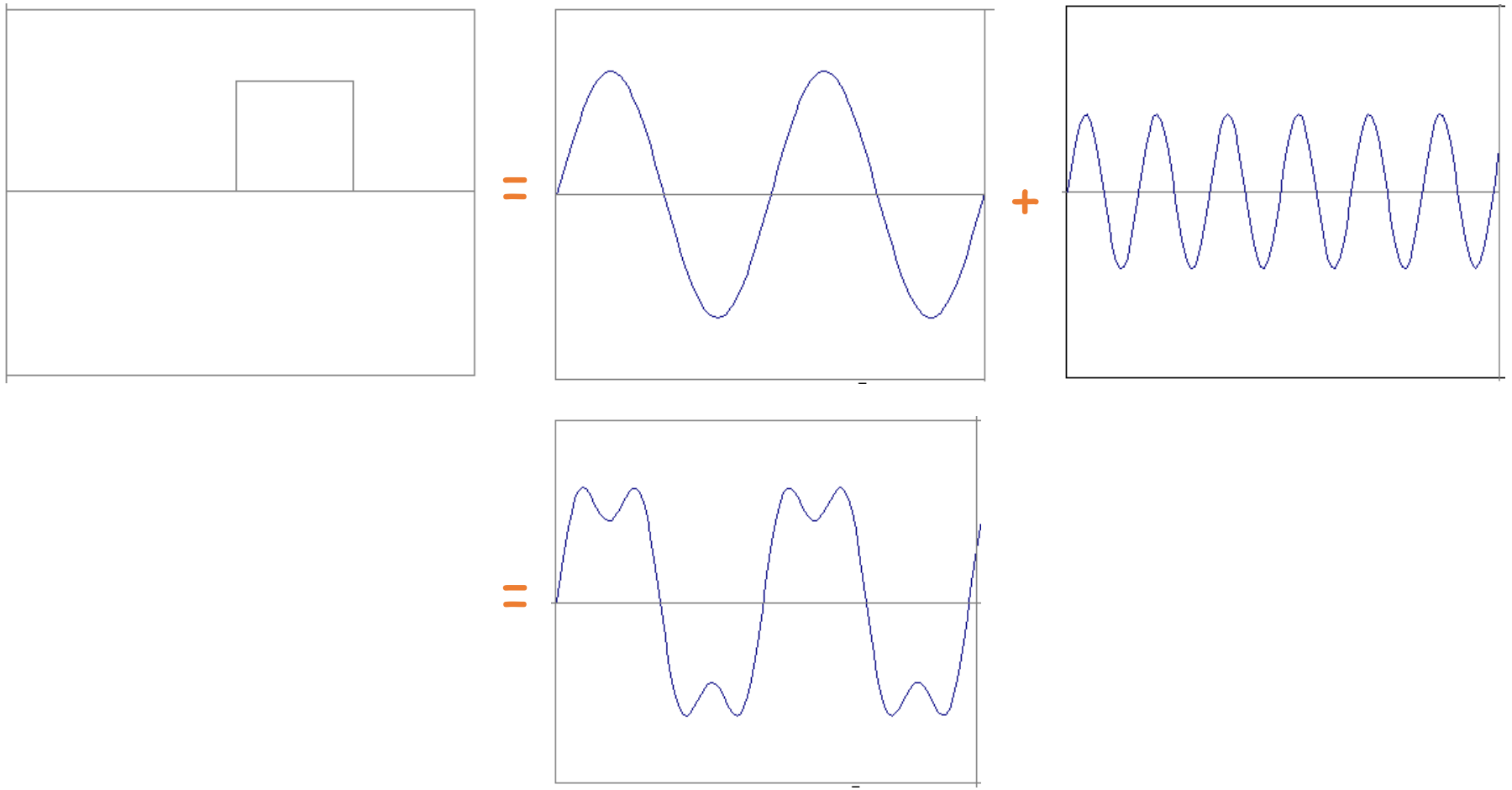


Frequency Spectra

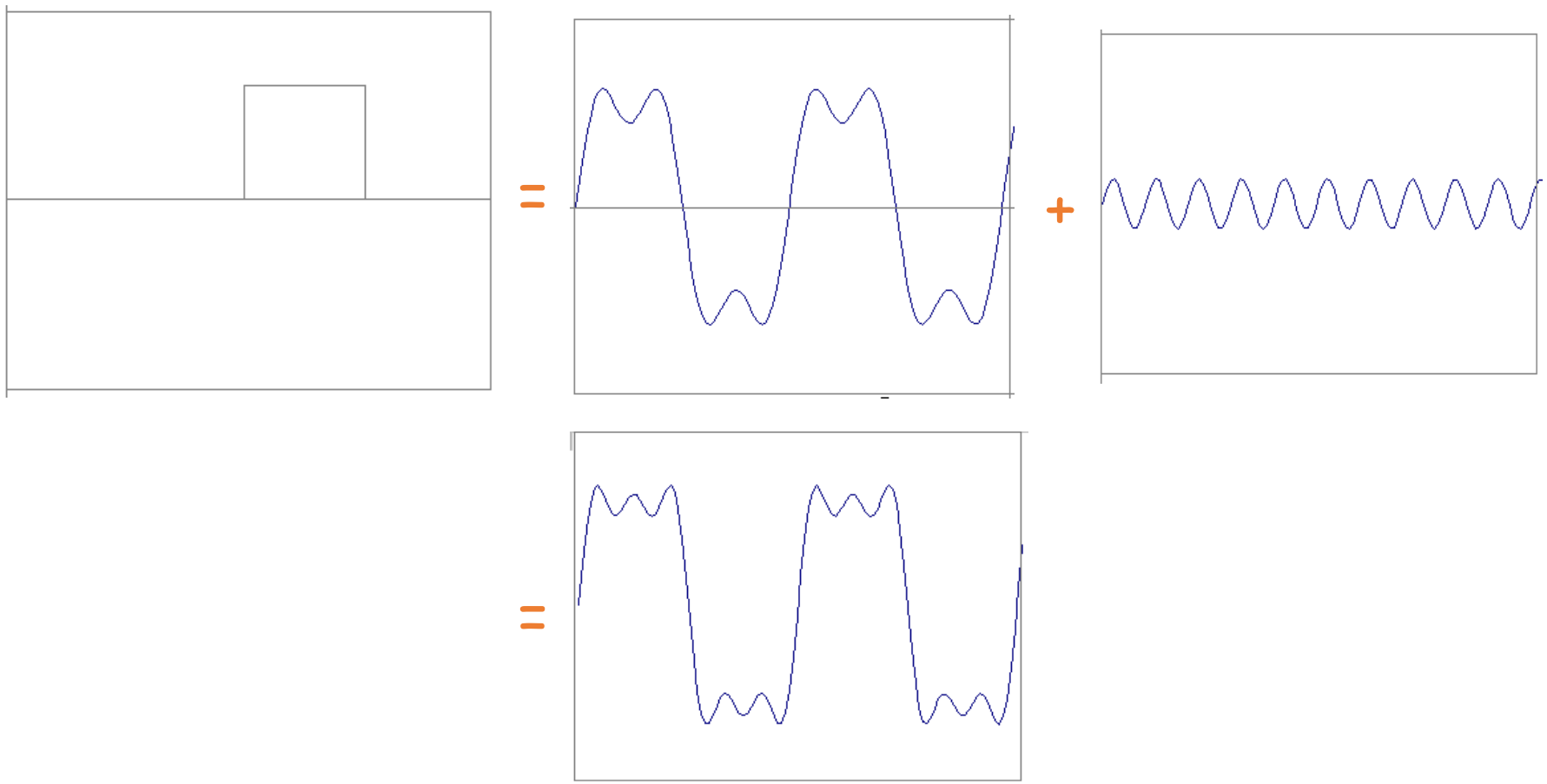
- Usually, frequency is more interesting than the phase



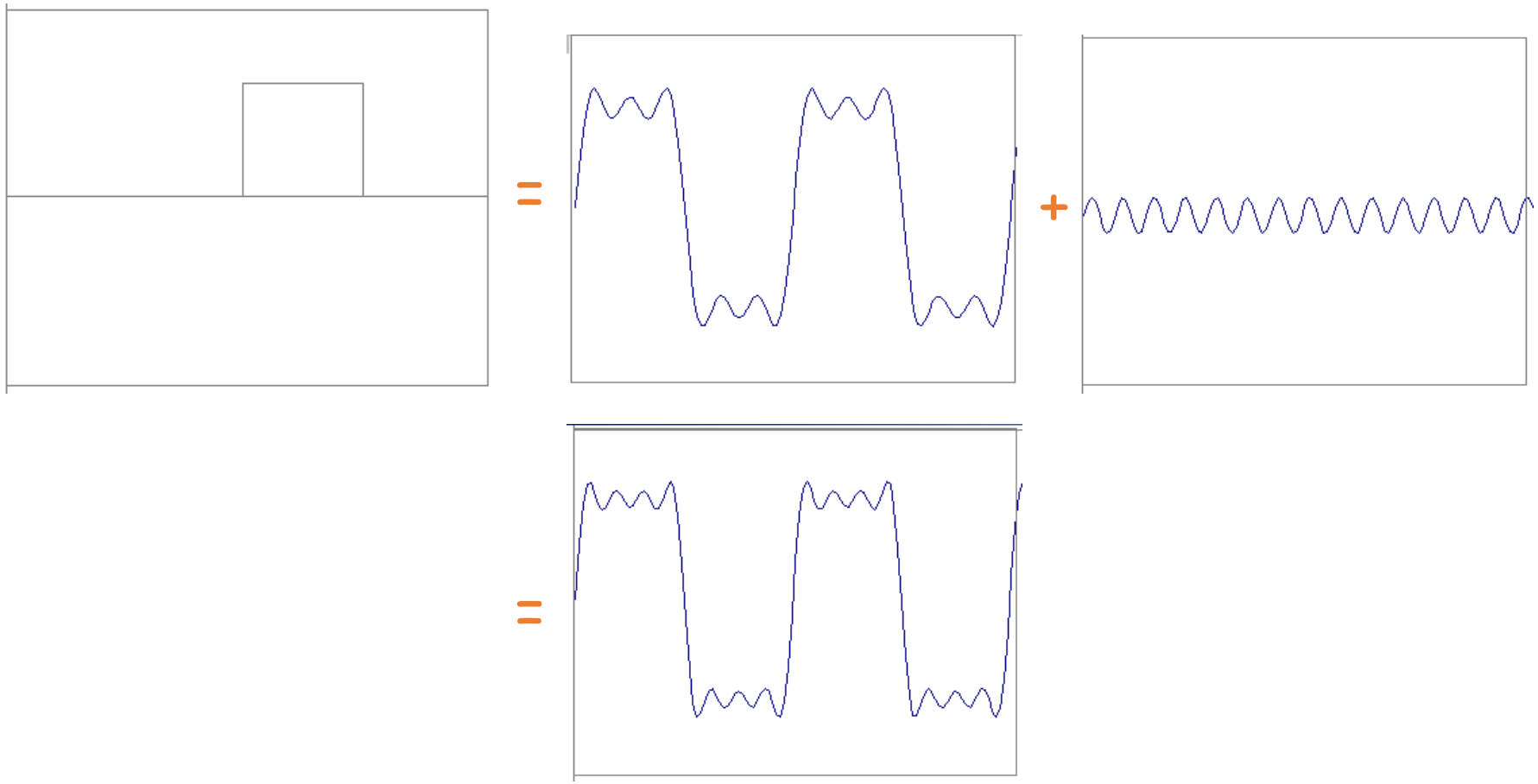
Frequency Spectra



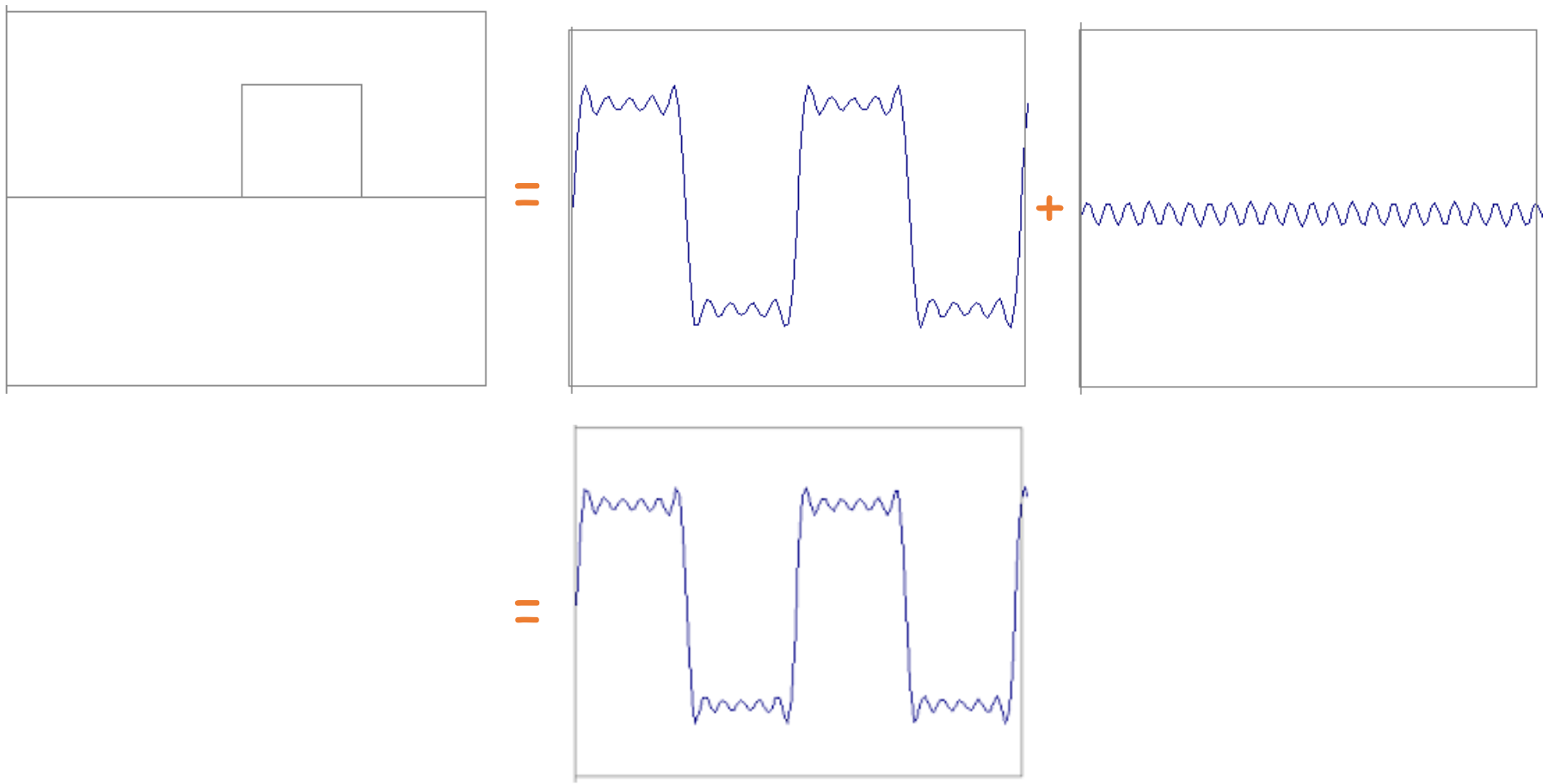
Frequency Spectra



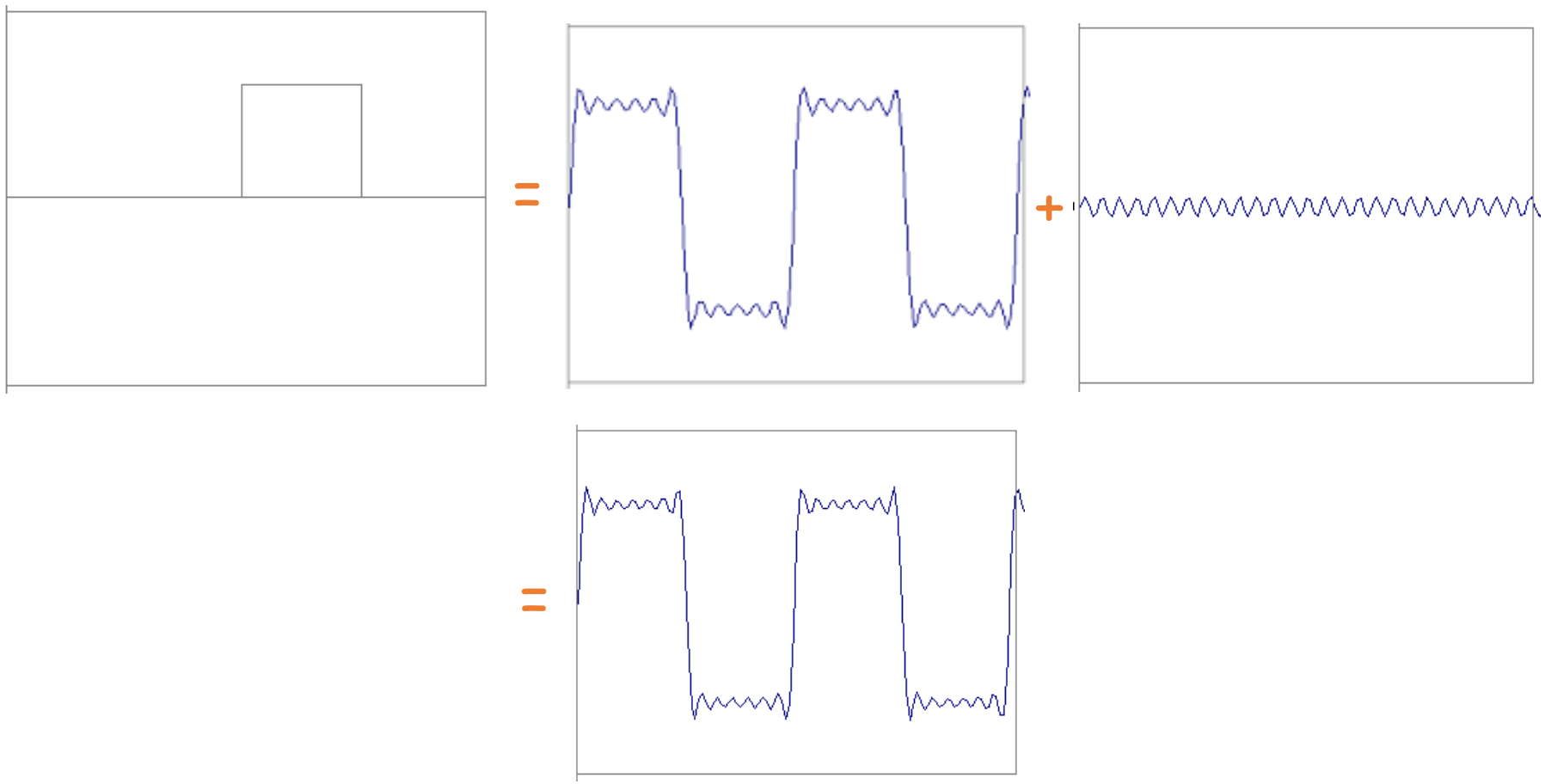
Frequency Spectra



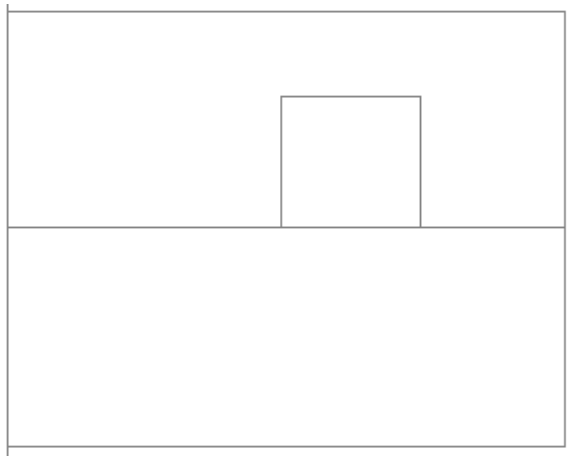
Frequency Spectra



Frequency Spectra

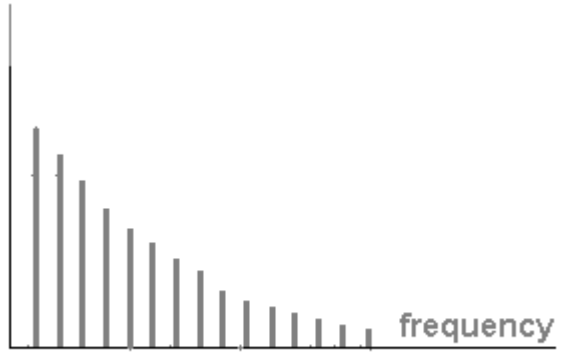


Frequency Spectra

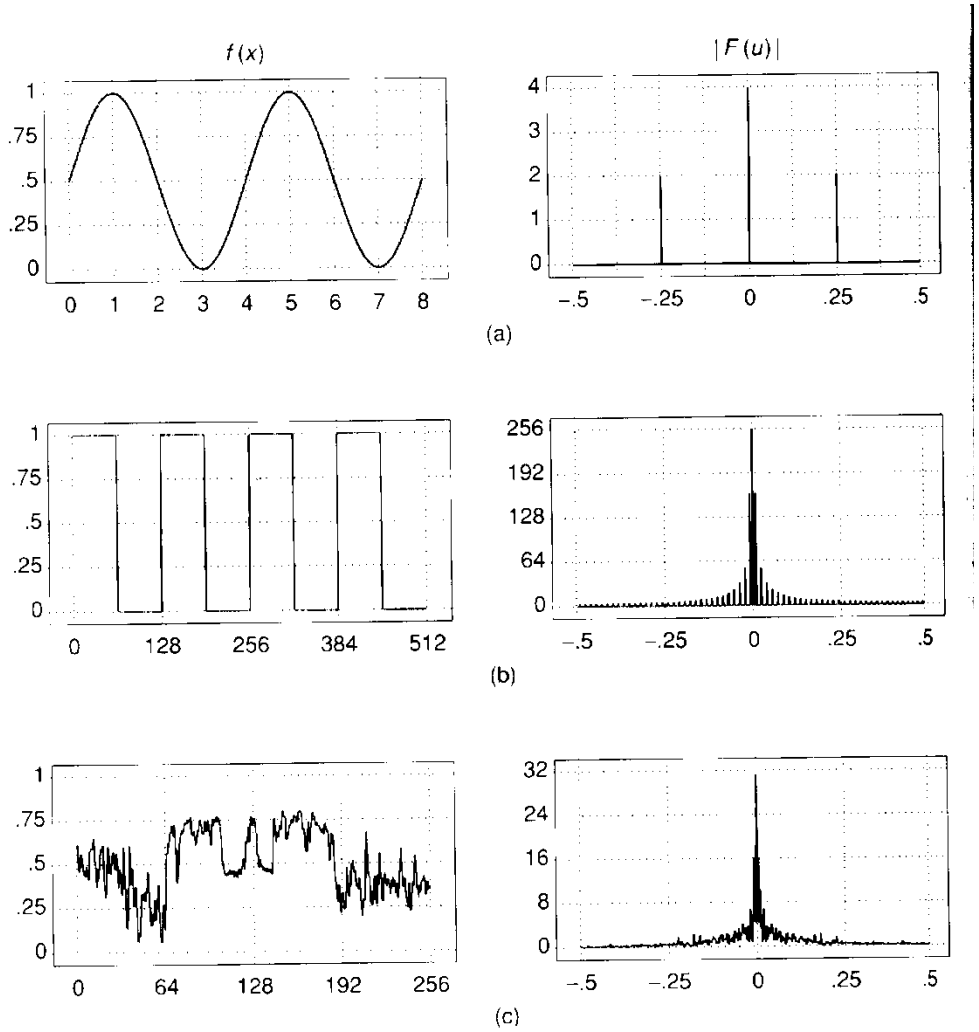


=

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

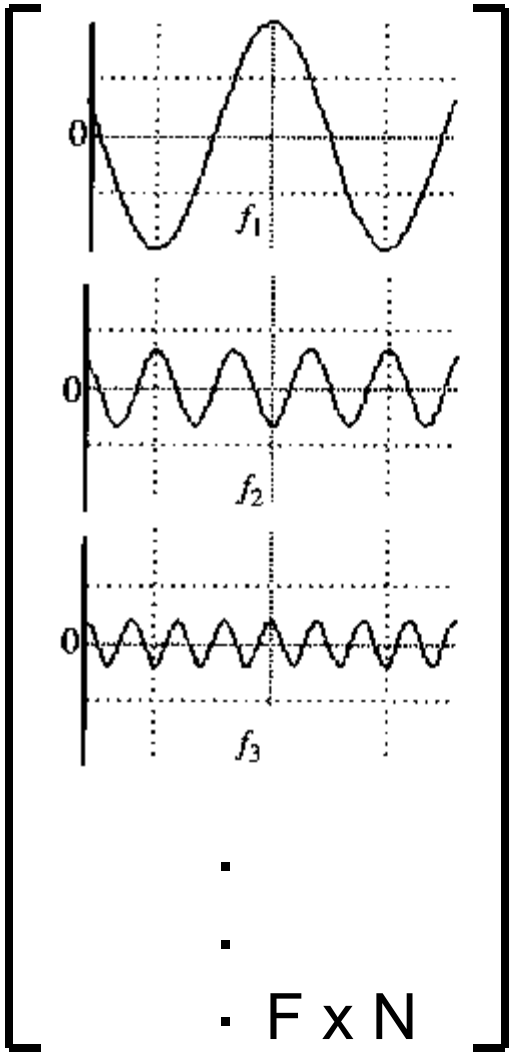


Frequency Spectra

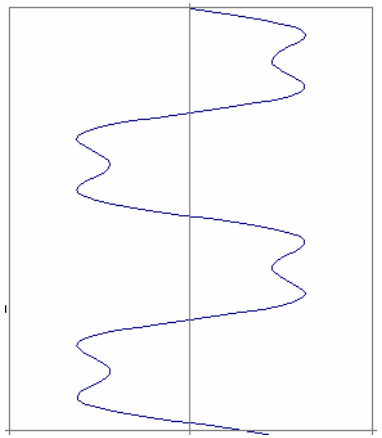


FT: Just a change of basis

$$M * f(x) = F(\omega)$$

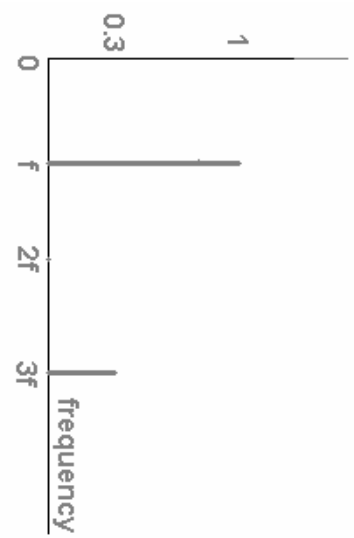


*



N x 1

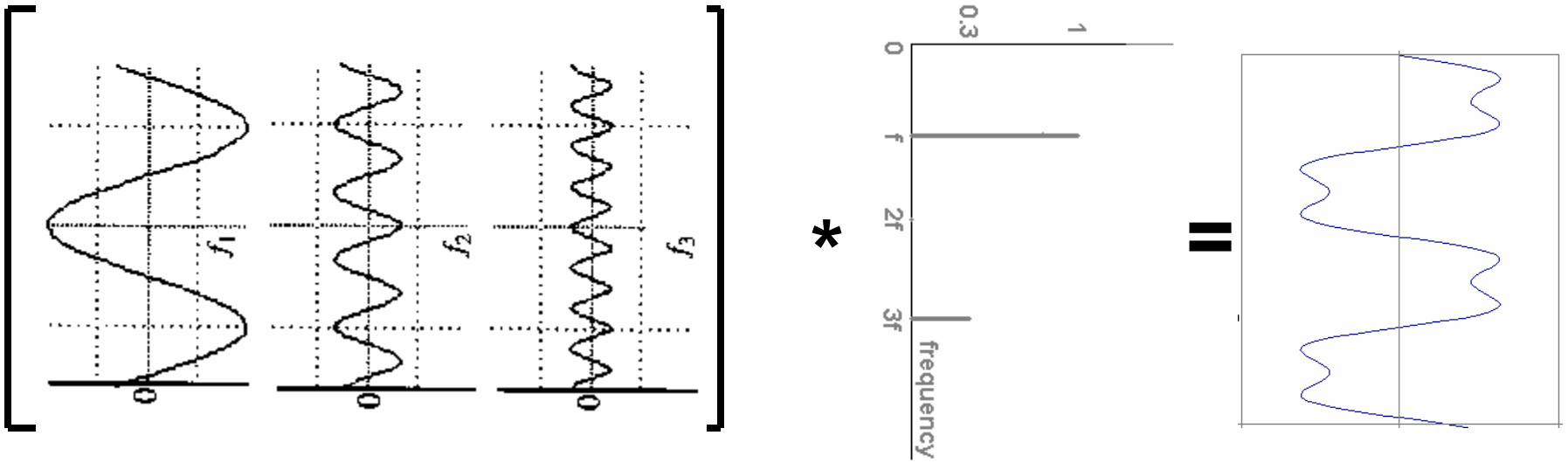
=



F x 1

IFT: Just a change of basis

$$M^{-1} * F(\omega) = f(x)$$



N x F F x 1 N x 1

Finally: Scary Math

$$\text{Fourier Transform}: \quad F(\omega) = \int_{-\infty}^{+\infty} f(x) e^{-i\omega x} dx$$

$$\text{Inverse Fourier Transform}: \quad f(x) = \frac{1}{2\pi} \int_{-\infty}^{+\infty} F(\omega) e^{i\omega x} d\omega$$

- …not really scary: $e^{i\omega x} = \cos(\omega x) + i \sin(\omega x)$
- is hiding our old friend: $\sin(\omega x + \phi)$

phase can be encoded by sin/cos pair →

$$P\cos(x) + Q\sin(x) = A\sin(x + \phi)$$

$$A = \pm\sqrt{P^2 + Q^2} \qquad \phi = \tan^{-1}\left(\frac{P}{Q}\right)$$

- So it's just our signal *f(x)* times sine at frequency ω

Discrete Fourier Transform

$$X[k] = \sum_{n=0}^{N-1} x[n] e^{-j 2\pi k n / N}.$$

Discrete Fourier Transform in 2D

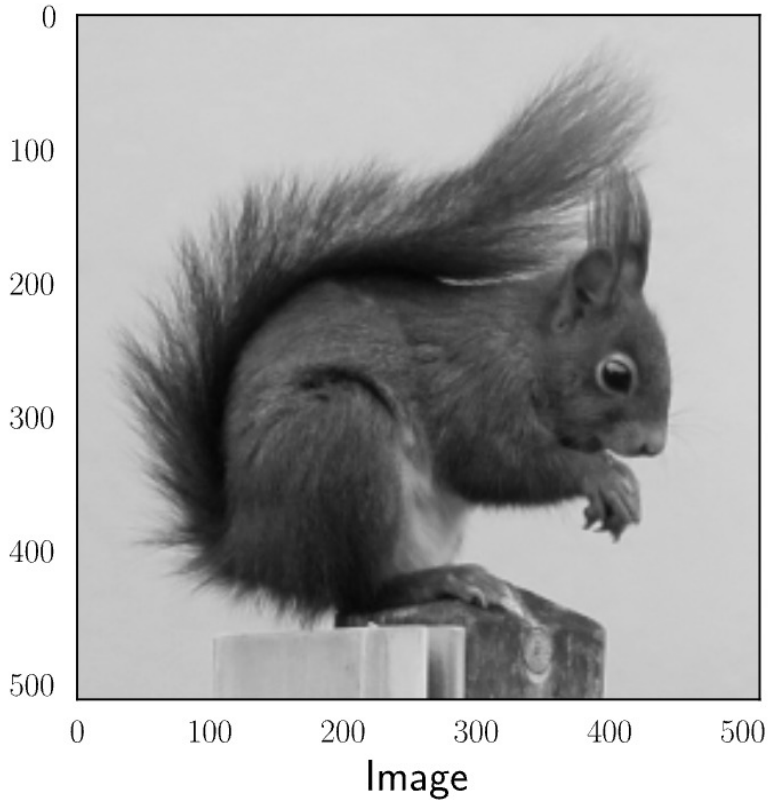
The discrete Fourier transform (DFT) of an image f of size $M \times N$ is an image F of same size defined as:

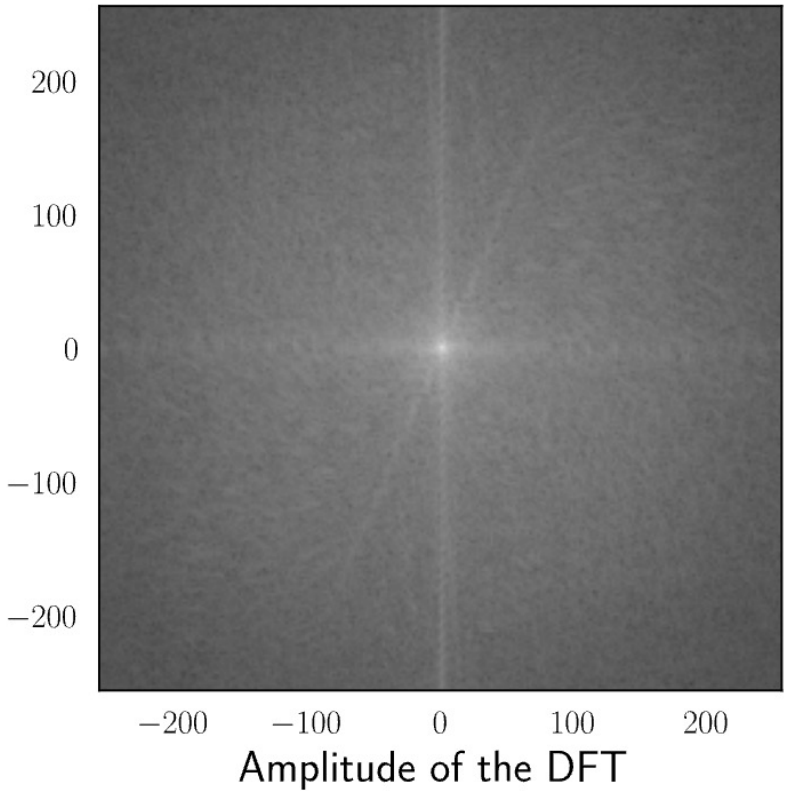
$$F(u,v) = \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} f(m,n) e^{-j\,2\pi \left(\frac{um}{M} + \frac{vn}{N}\right)}$$

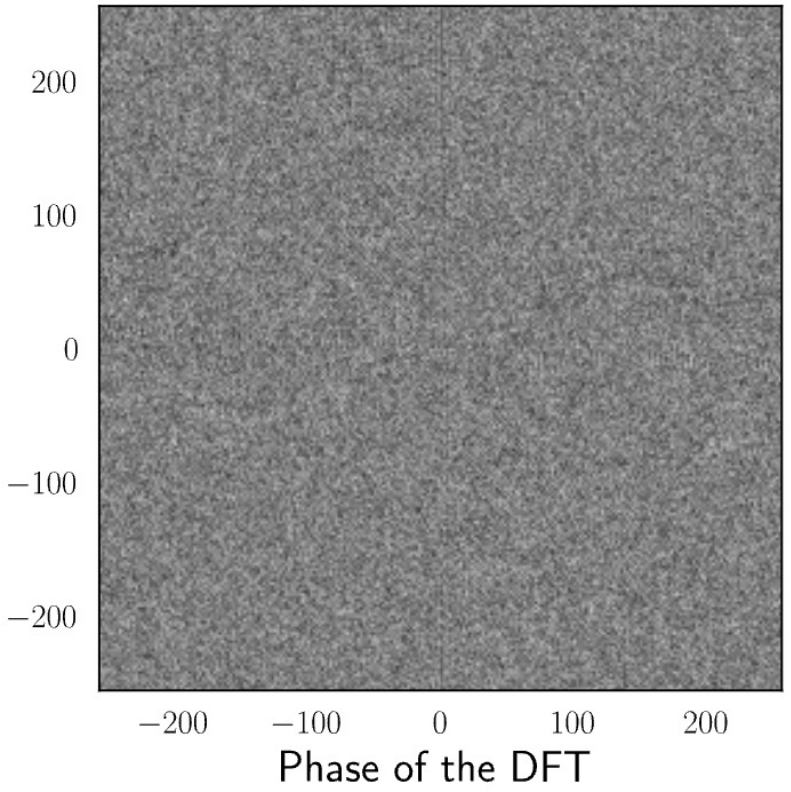
Today's Lecture

- Edge Detection
- Fourier Analysis (in 1D)
- Fourier Analysis (in 2D)
- Applications of Fourier Analysis
  - Hybrid Image
  - JPG Compression

Discrete Fourier Transform (DFT) in 2D

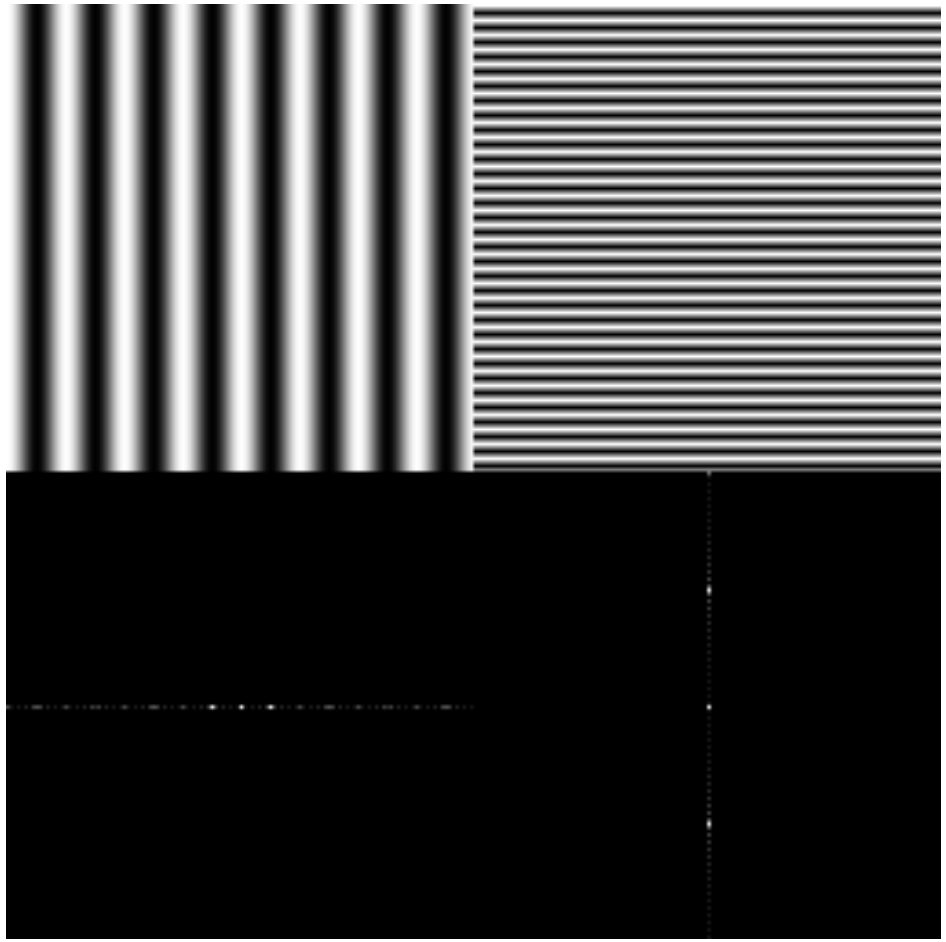


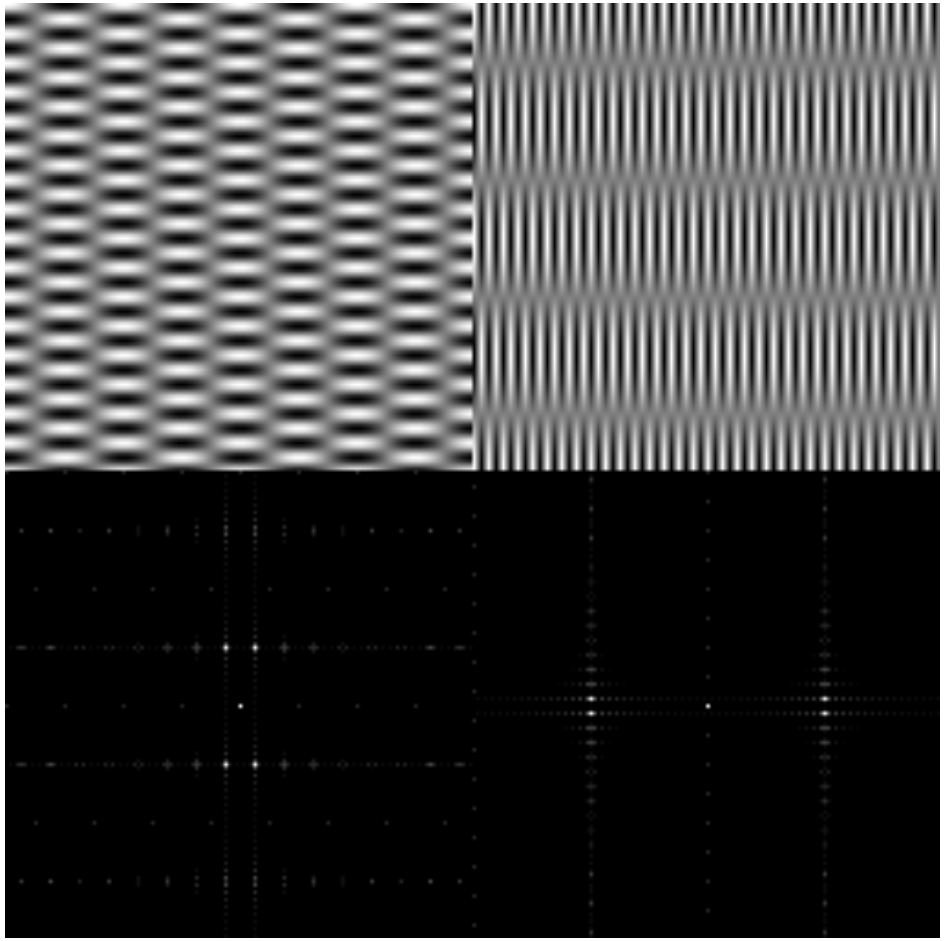




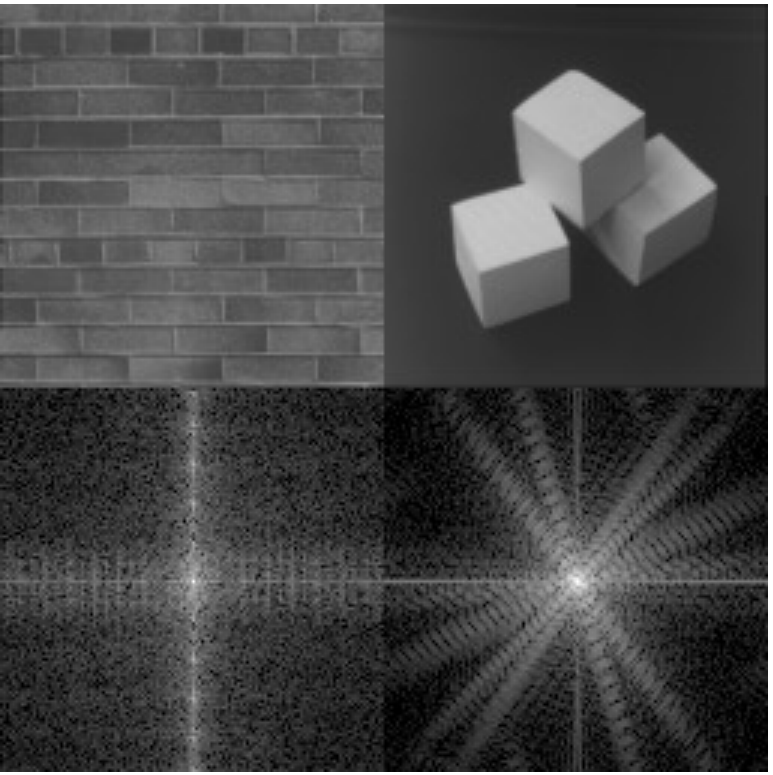
Credits: https://vincmazet.github.io/bip/filtering/fourier.html

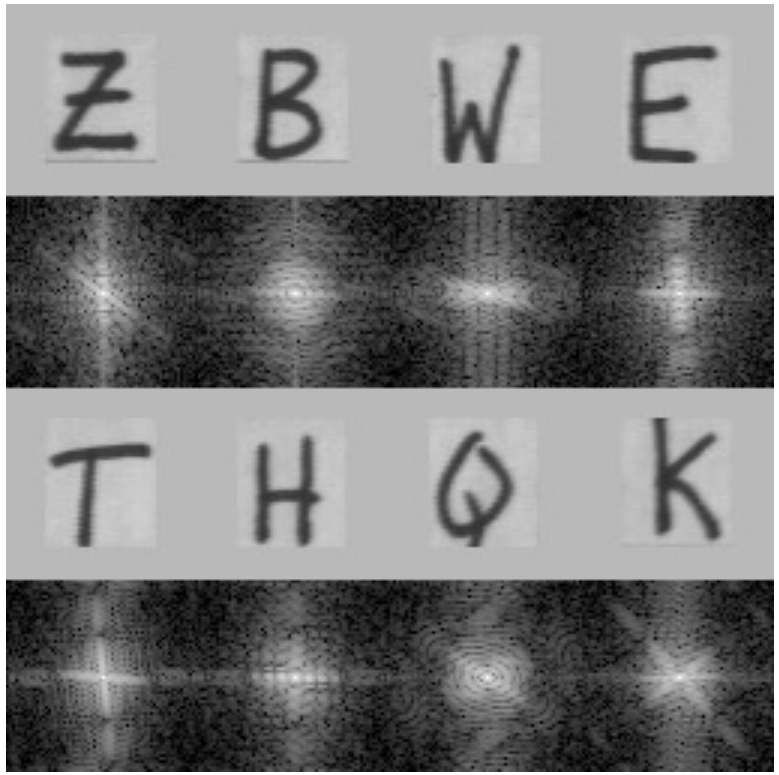
Amplitude Spectrum of DFT

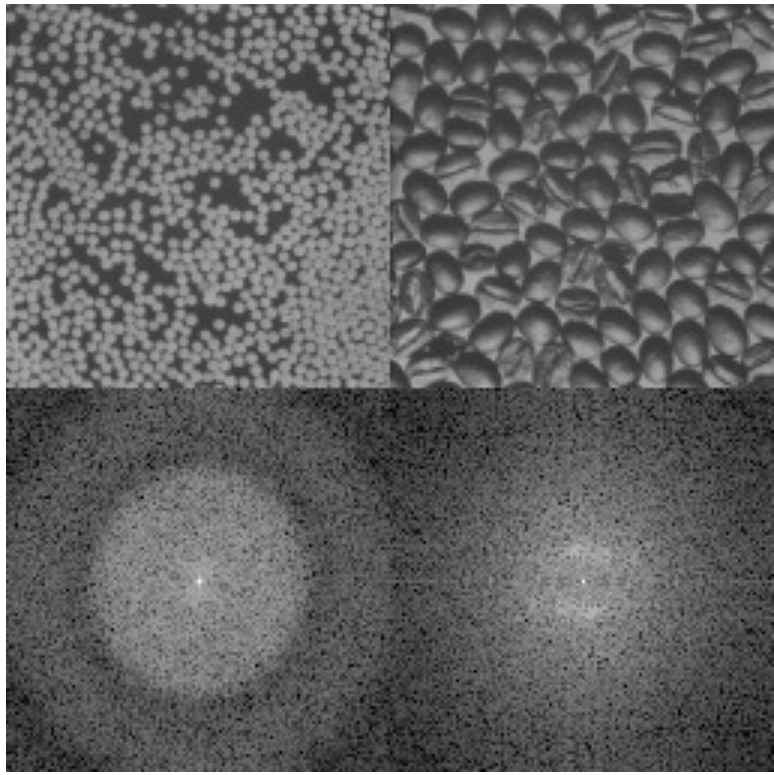




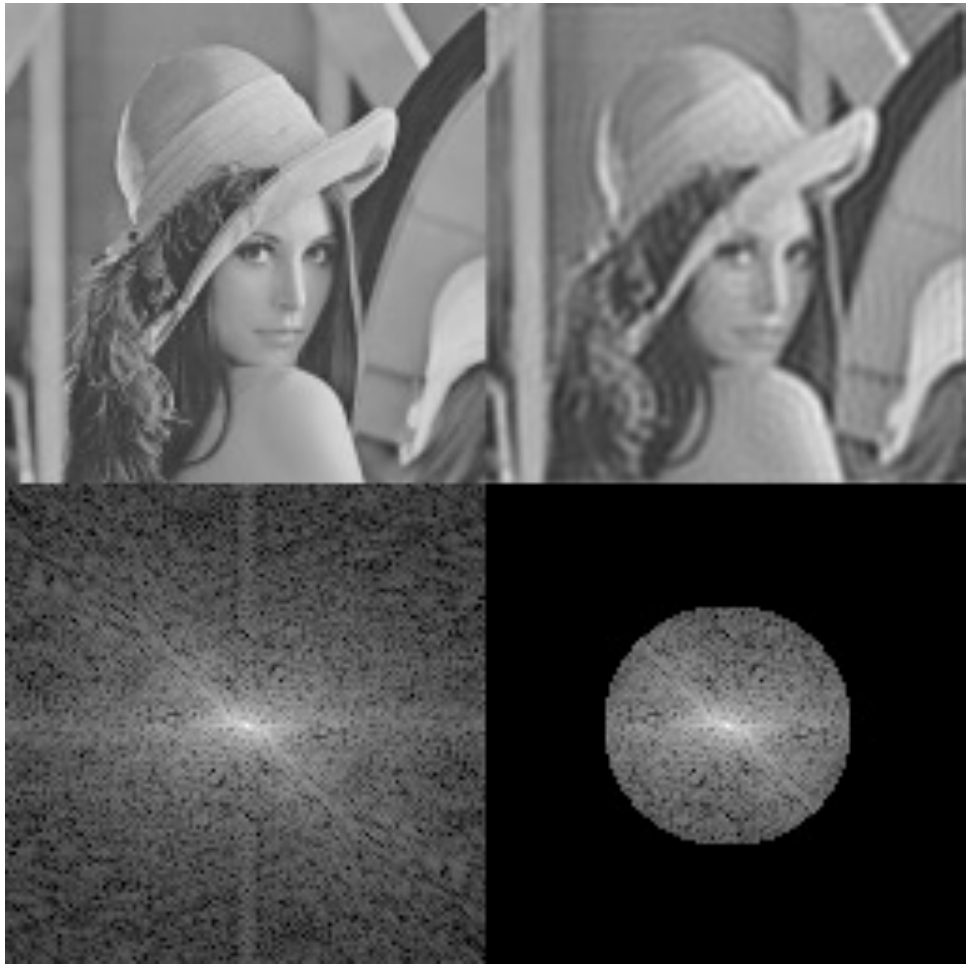
Amplitude Spectrum of DFT







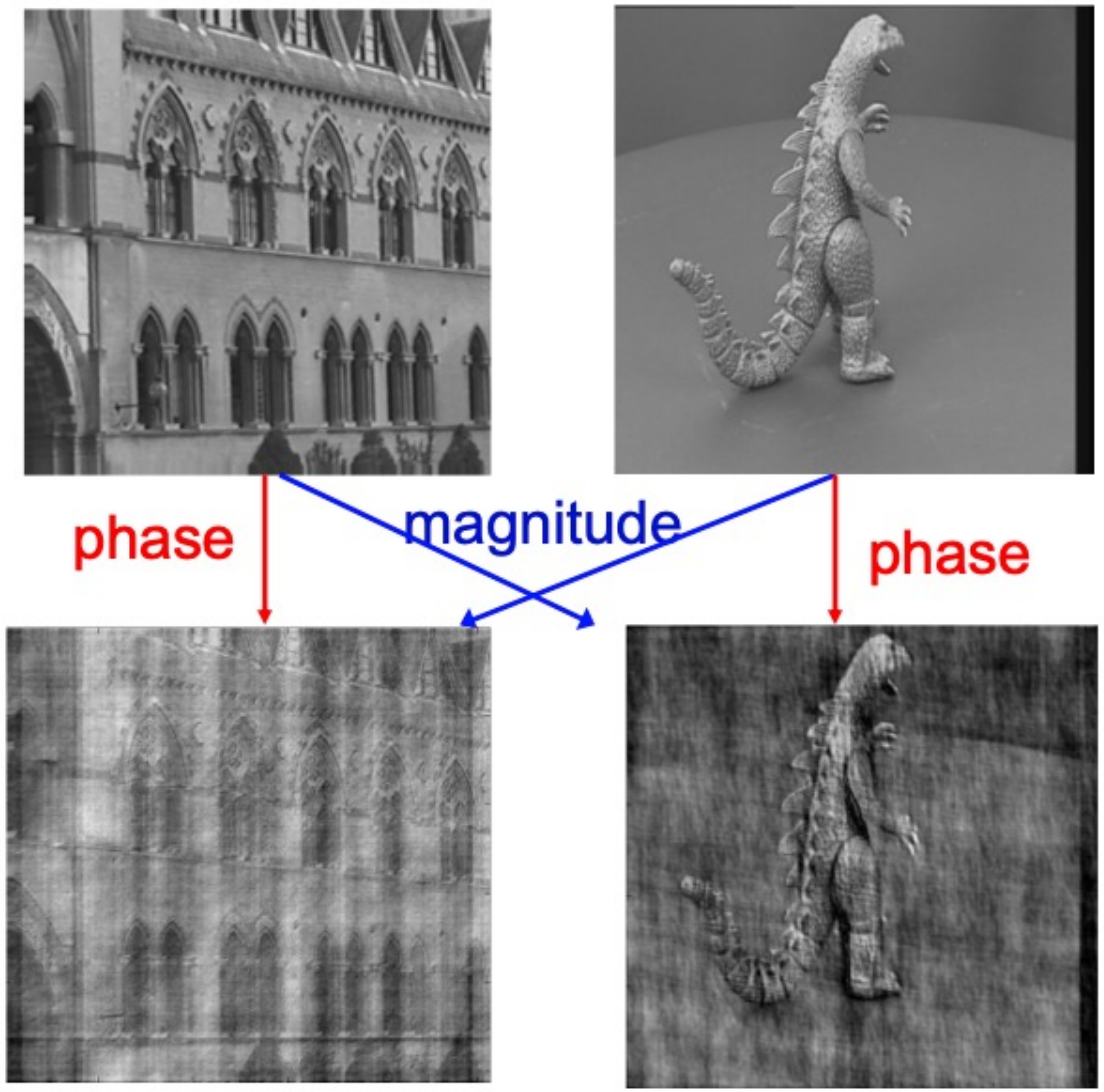
Filtering & Frequency Domain



Low pass Filtering with Gaussian Filter

High pass Filtering with Laplacian Filter

The importance of Phase



Slide: Andrew Zisserman

The Convolution Theorem

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

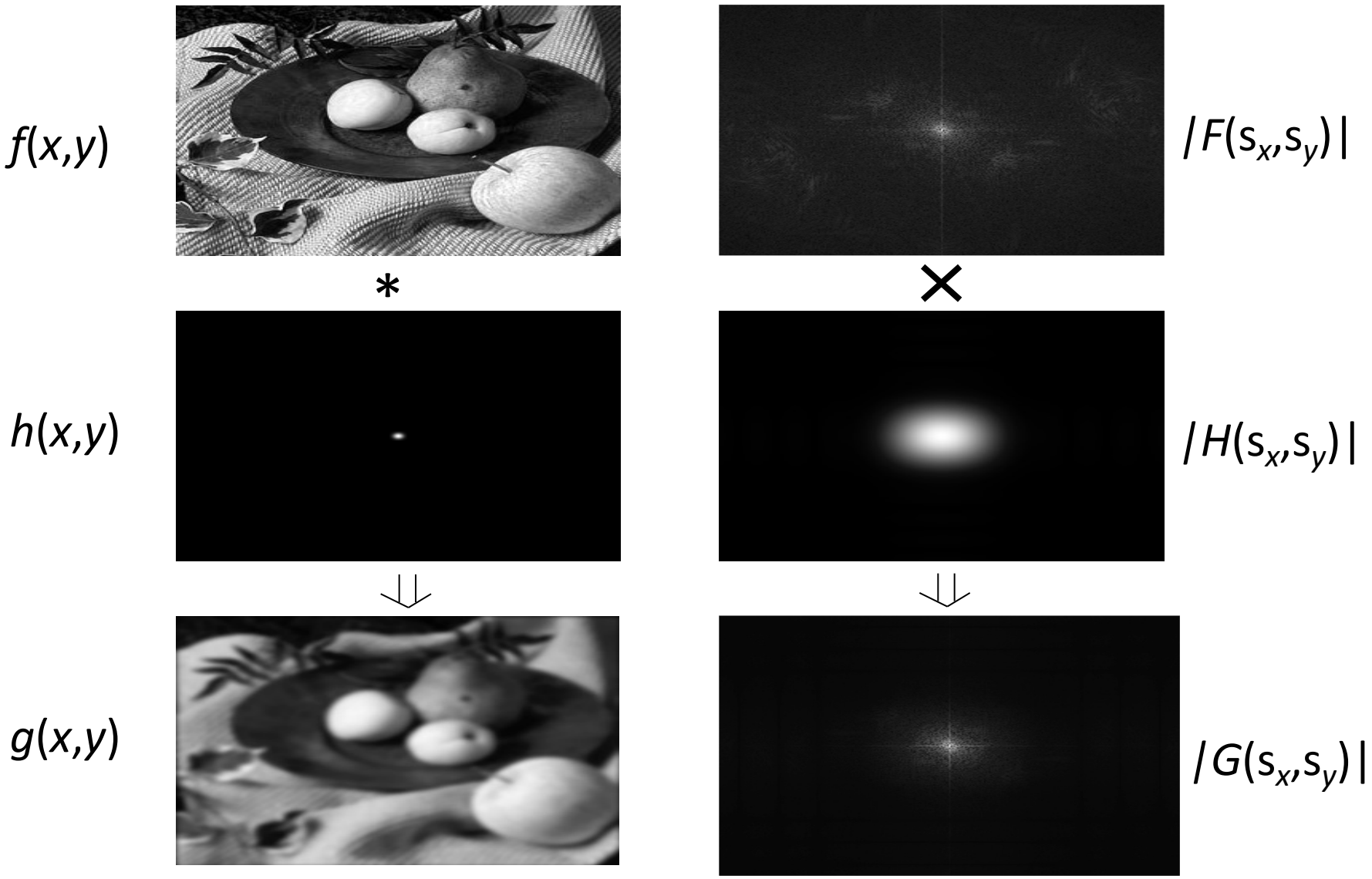
$$\mathrm{F}[g * h] = \mathrm{F}[g]\mathrm{F}[h]$$

- 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!

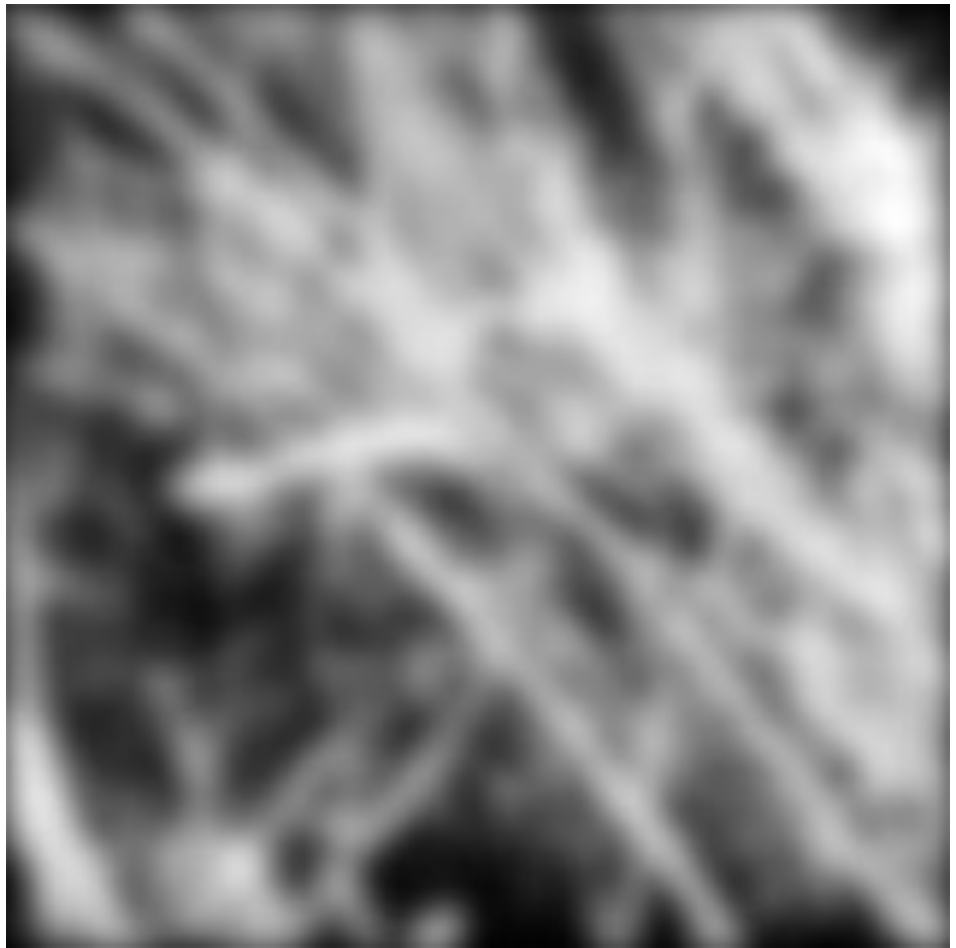
2D convolution theorem example



Filtering

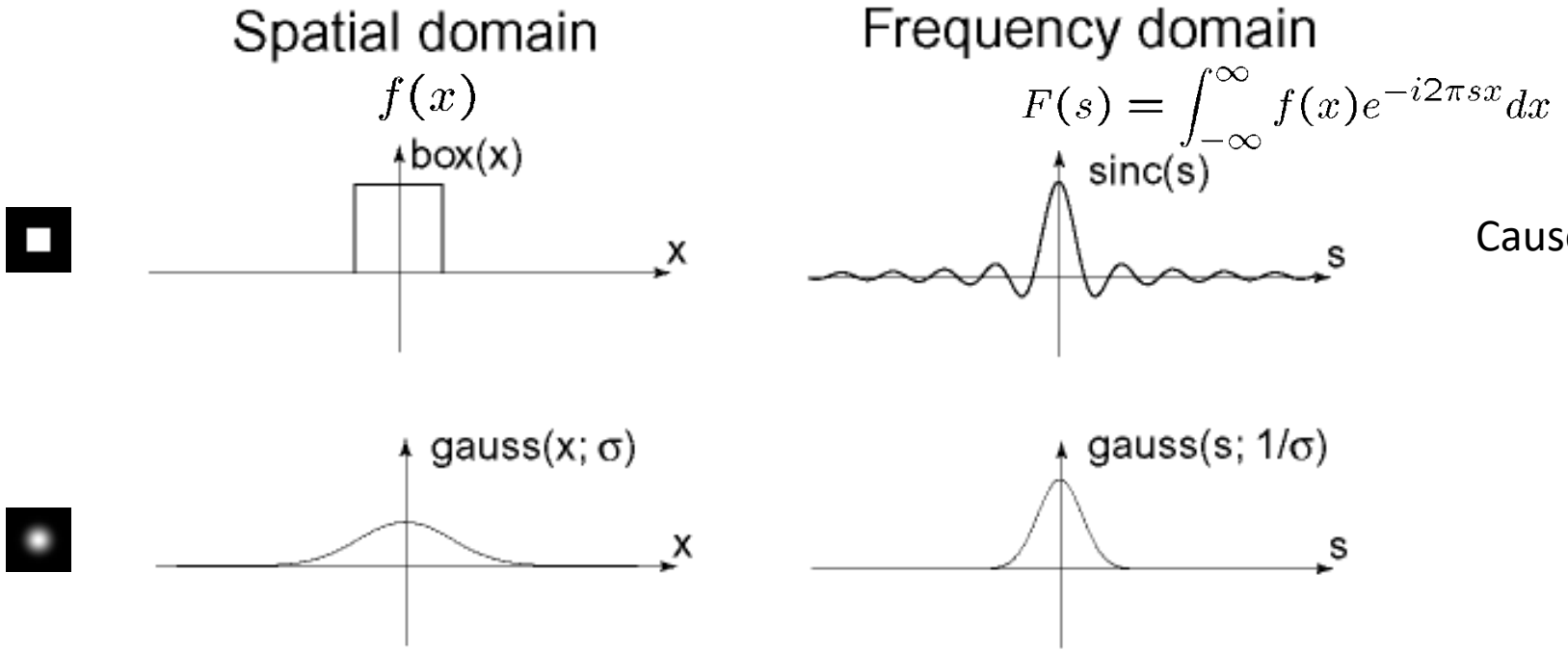
Why does the Gaussian give a nice smooth image, but the square filter give edgy artifacts?

Gaussian 



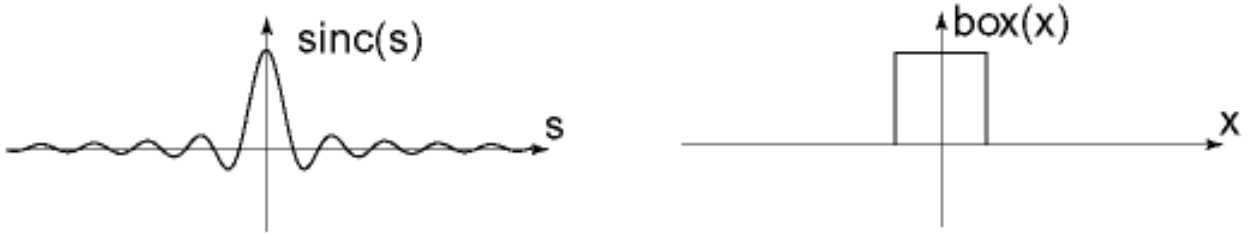
Box filter 

Fourier Transform pairs

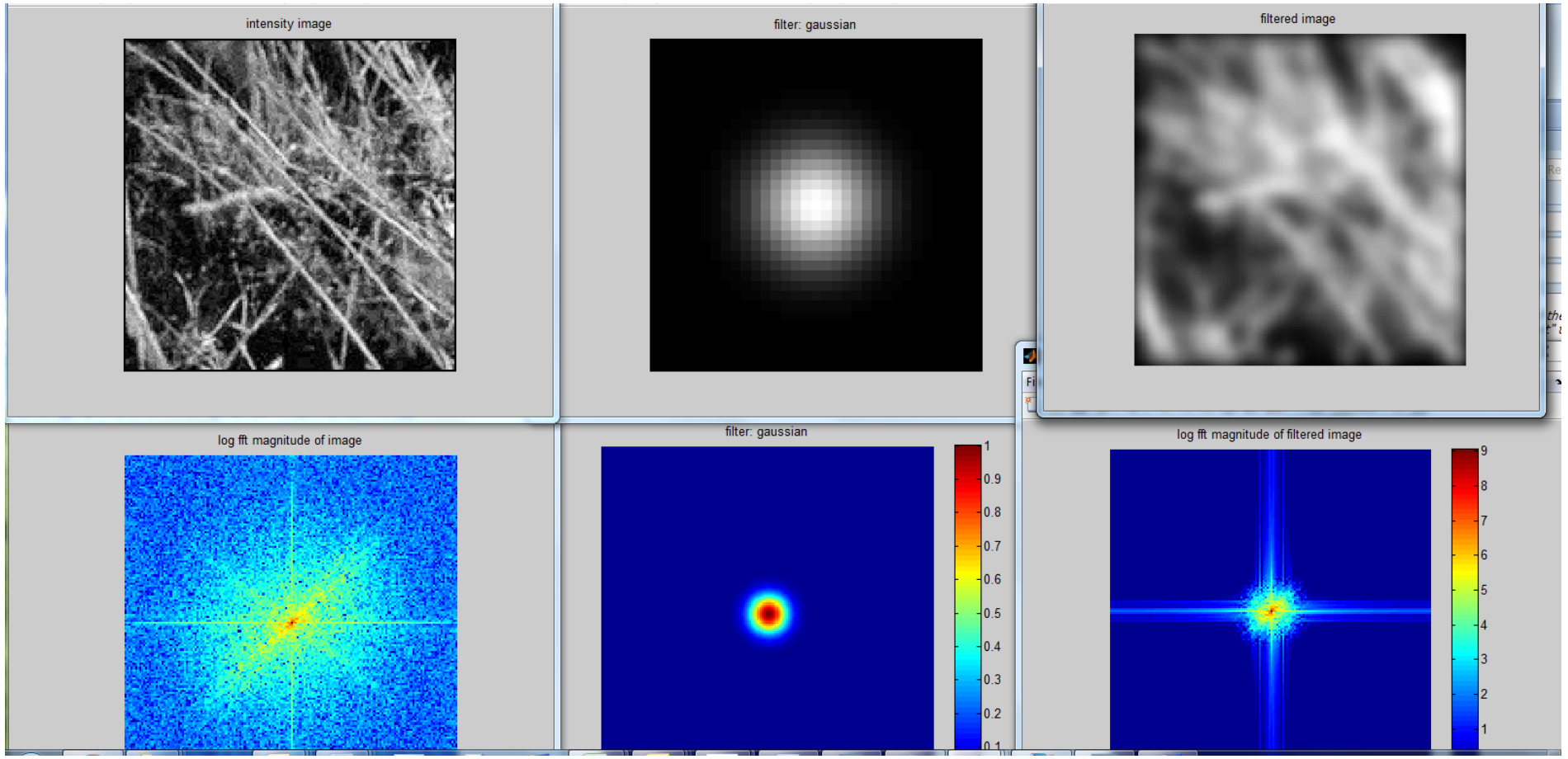


Causes aliasing!

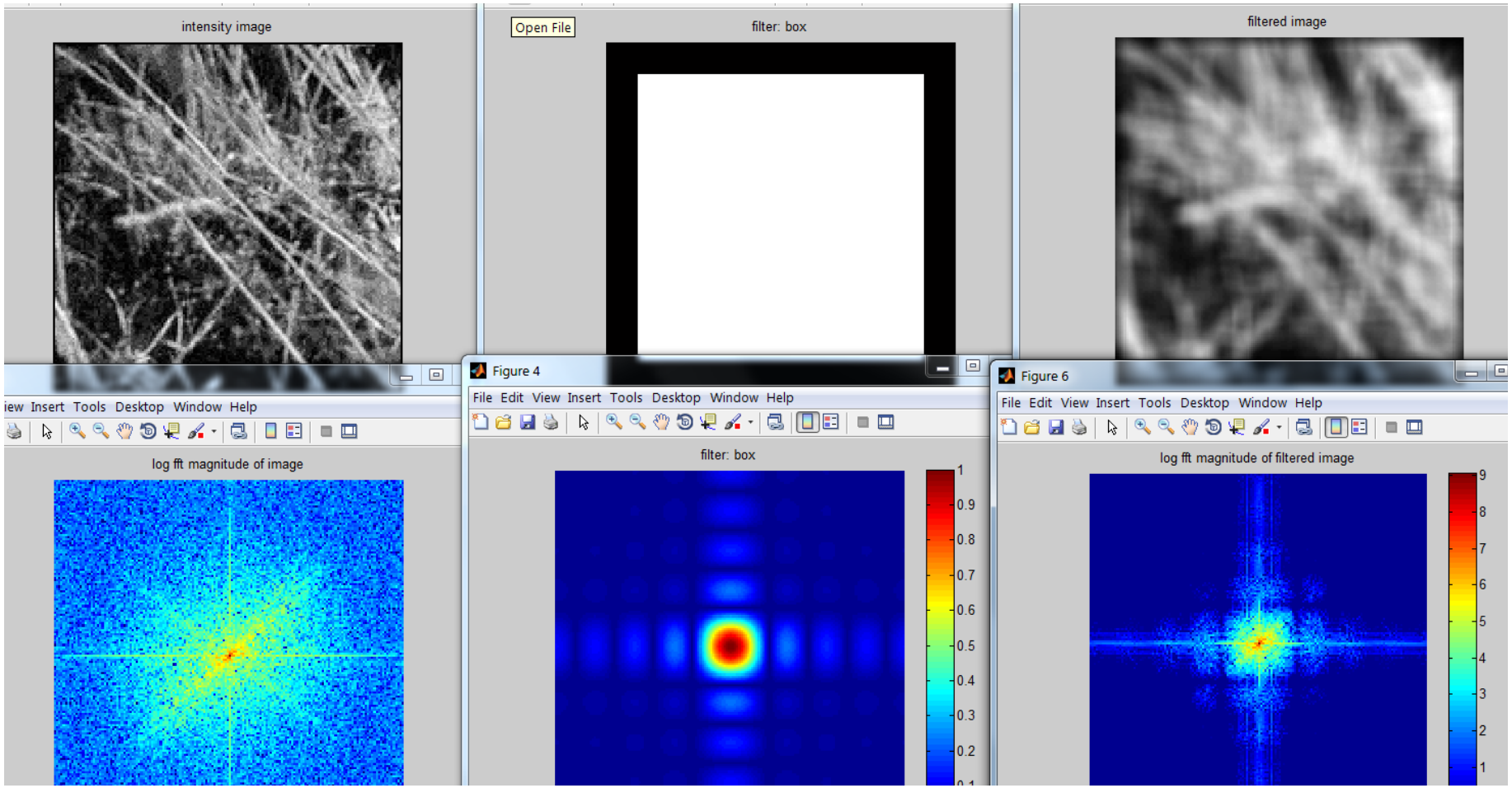
What is the best low pass filter?



Gaussian


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

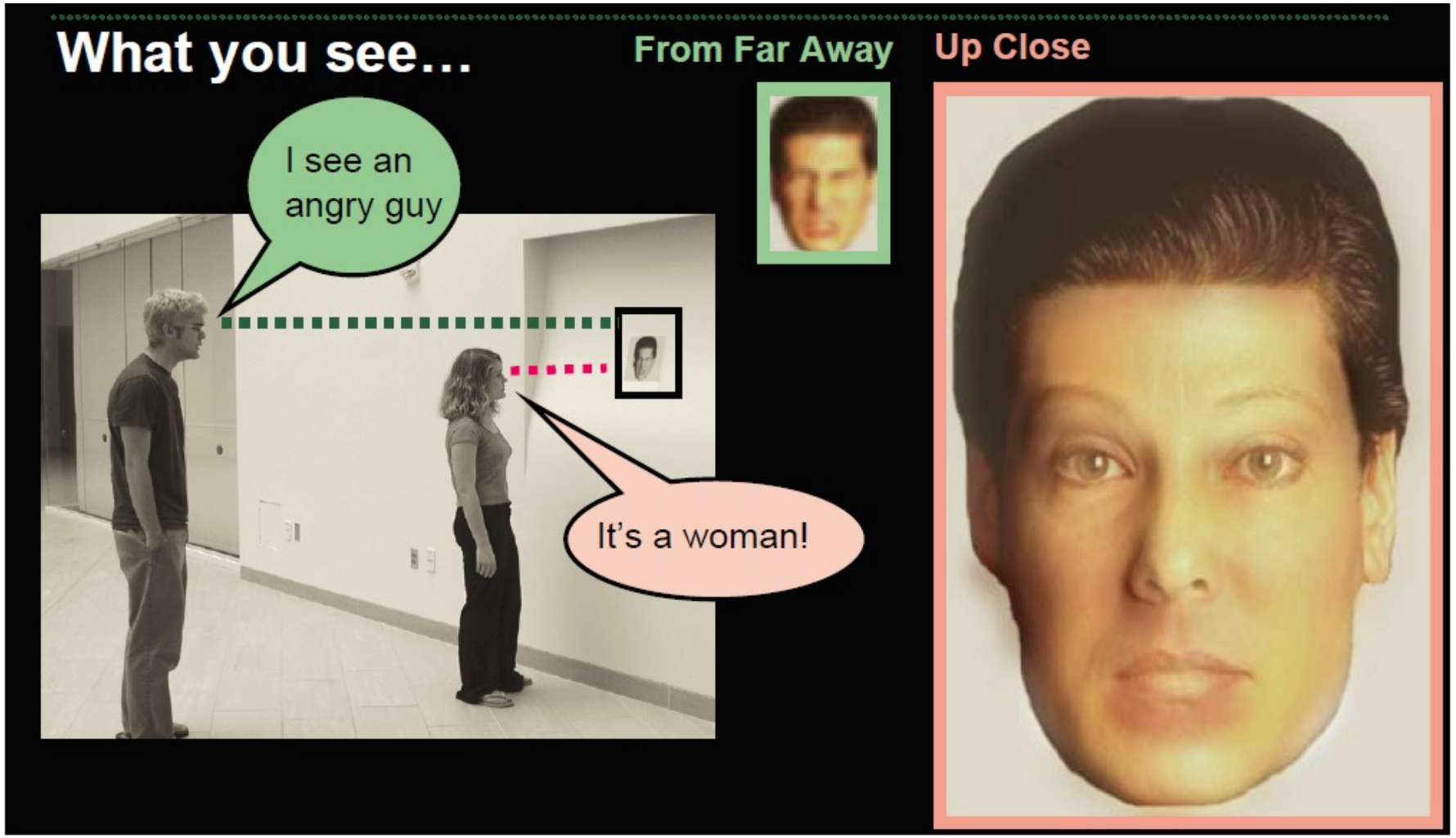
Box Filter


intensity image
Open File
filter: box
filtered image
iew Insert Tools Desktop Window Help
log fft magnitude of image
Figure 4
File Edit View Insert Tools Desktop Window Help
filter: box
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
Figure 6
File Edit View Insert Tools Desktop Window Help
log fft magnitude of filtered image
9
8
7
6
5
4
3
2
1

Today’s Lecture

- Edge Detection
- Fourier Analysis (in 1D)
- Fourier Analysis (in 2D)
- Applications of Fourier Analysis
  - Hybrid Image
  - JPG Compression

Application: Hybrid Images (in HW)



Aude Oliva & Antonio Torralba & Philippe G Schyns, SIGGRAPH 2006

Application: Hybrid Images

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

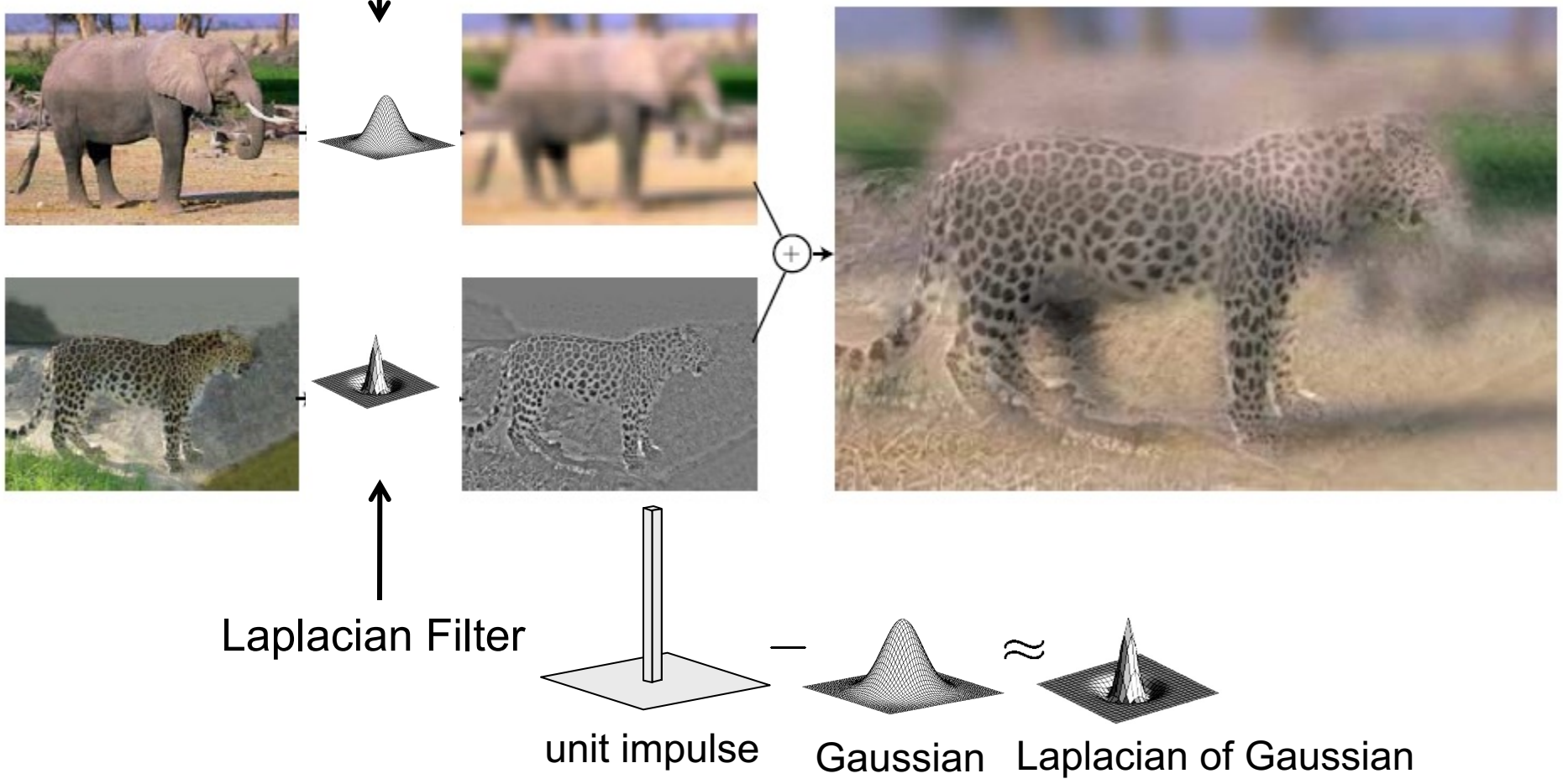
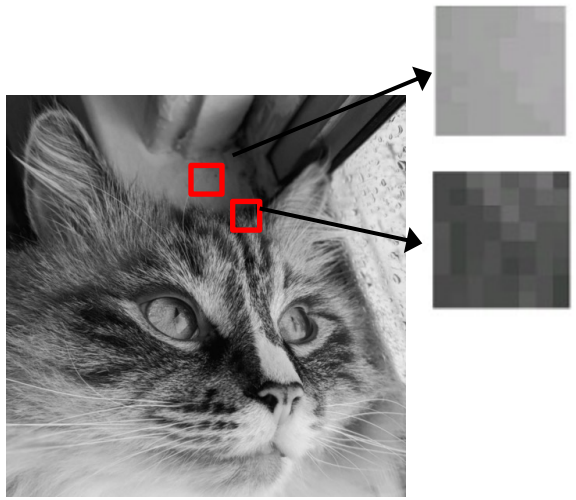


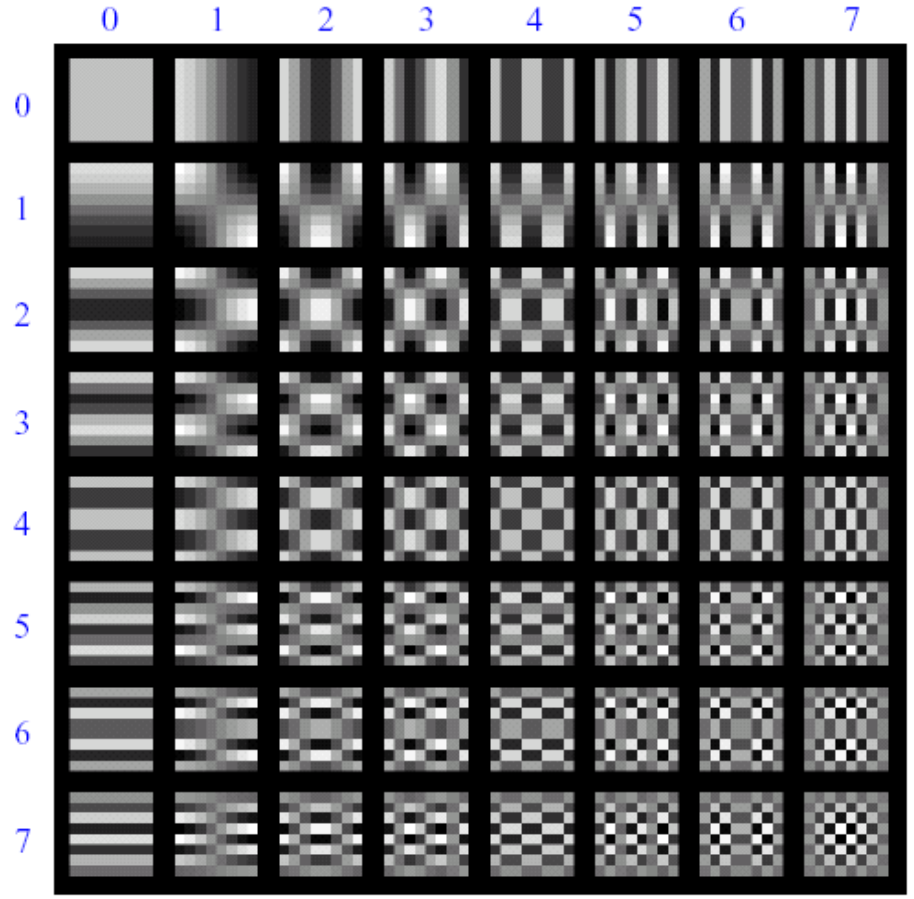
Image Compression

89k

Lossy Image Compression (JPEG)



cut up into 8x8 blocks



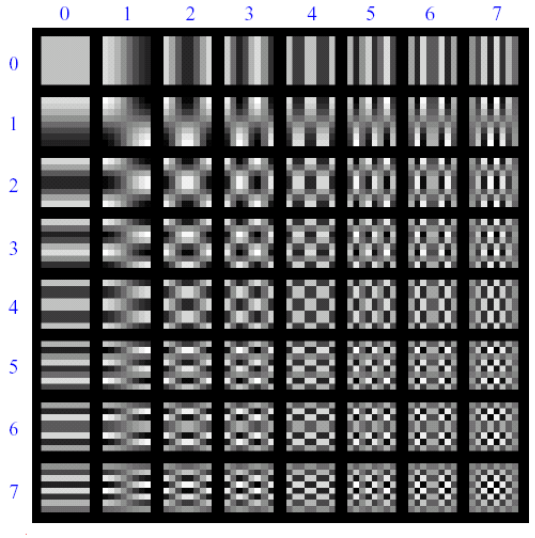
For each block

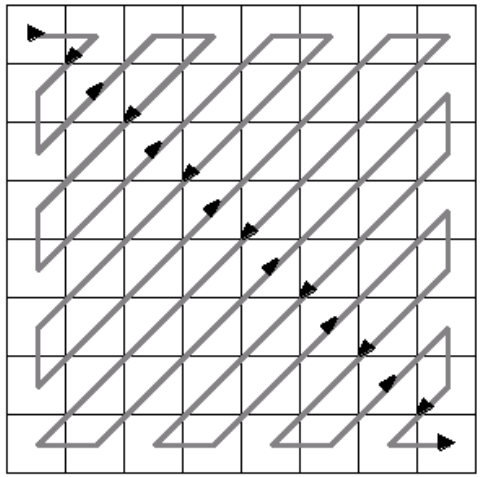
- Compute DCT coefficients.
- Coarsely quantize
  - Many high frequency components will become zero
- Encode (e.g., with Huffman coding)

Block-based Discrete Cosine Transform (DCT)

Using DCT in JPEG

- The first coefficient $B(0,0)$ is the DC component, the average intensity
- The top-left coeffs represent low frequencies, the bottom right – high frequencies





- DCT is very similar to DFT (Fourier Transform), but instead of using complex exponentials (e^{-jx}) we use real-valued cosine functions.
- A signal's DCT representation tends to have more of its energy concentrated in a small number of coefficients compared DFT, thus more suitable for compression.

JPEG compression comparison

89k

12k

Slide Credits

- CS5670, Introduction to Computer Vision, **Cornell Tech, by Noah Snavely.**
- CS 194-26/294-26: Intro to Computer Vision and Computational Photography, **UC Berkeley, by Alyosha Efros.**
- **CS 15-463, 663, 862, CMU, by Computational Photography, Ioannis Gkioulekas.**

Suggested Reading

- Fourier Transform in 5 minutes (video)
- Szeliski, Chapter 3.1, 3.2, 3.3, 3.4, 3.5
- Forsyth & Ponce, Chapter 4, Chapter 5.1, 5.2, 5.3