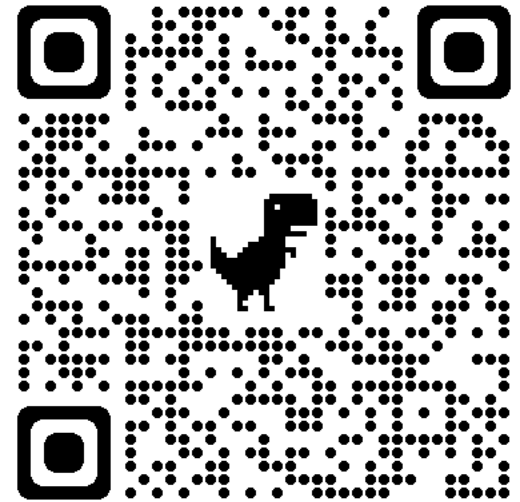


Lecture 6: Image Processing (cont.)

COMP 590/776: Computer Vision

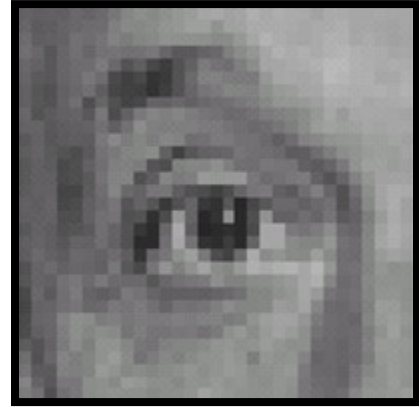
Instructor: Soumyadip (Roni) Sengupta

TA: Mykhailo (Misha) Shvets



Course Website:
Scan Me!

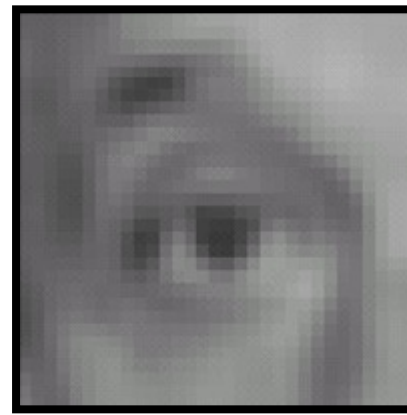
Recap



Original



Linear Filtering:
Cross-correlation
& Convolution



Blur



Shifted left by 1 pixel



Sharpening

Filters

Many Options:

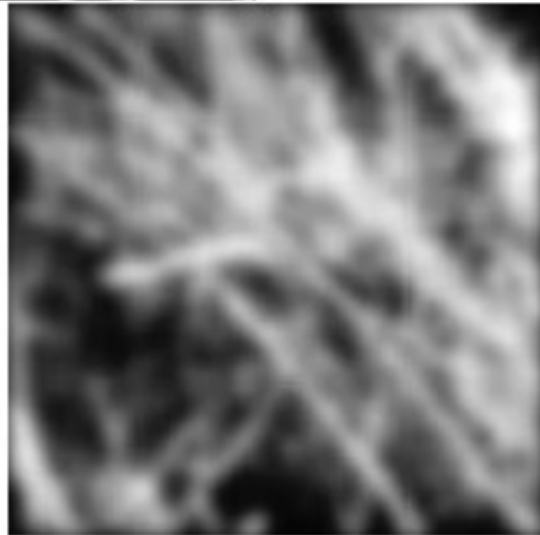
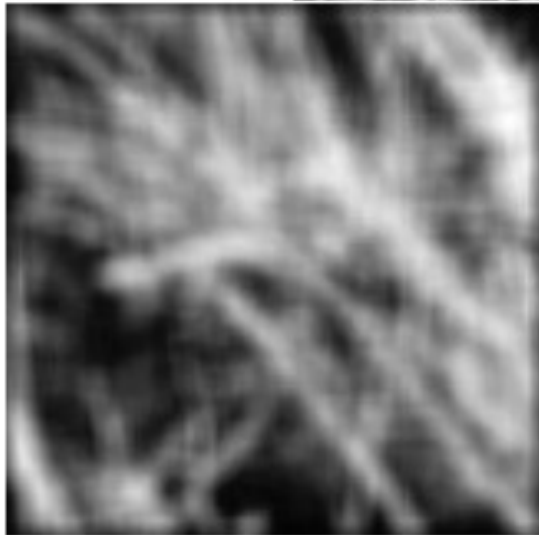
- Mean Filtering
- Gaussian Filtering
- Bilateral Filtering



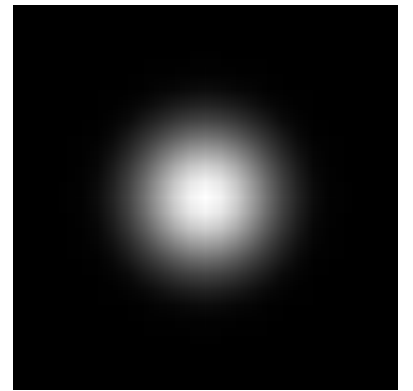
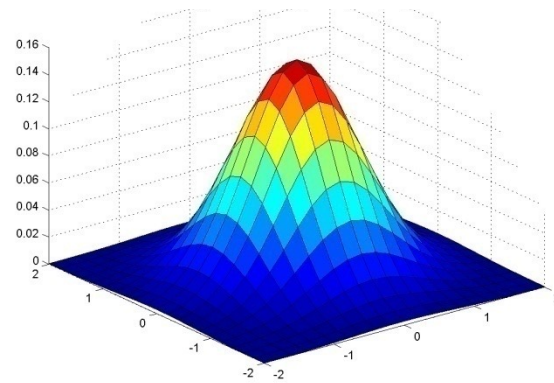
0	0	0
1	0	0
0	0	0

Laplacian Filtering

Mean vs. Gaussian filtering



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

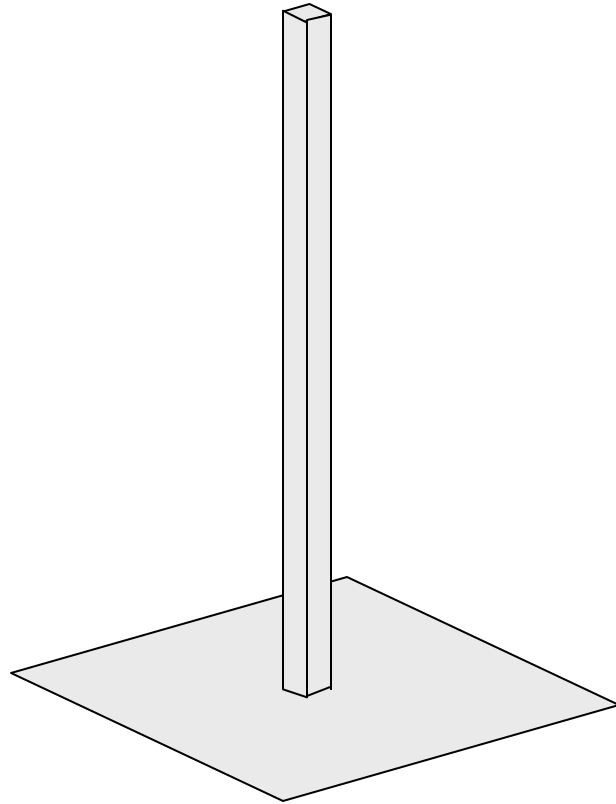


Sharpen filter

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

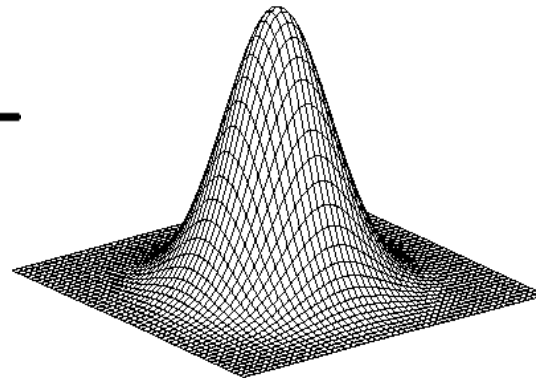
↑ image

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



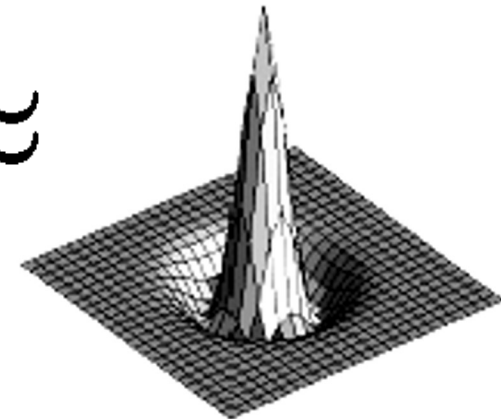
scaled impulse

−



Gaussian

≈



Sharpen filter

Edge Aware Smoothing: Gaussian vs Bilateral

Smooths everything nearby
(even edges)



Smooths everything nearby
with similar intensity



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

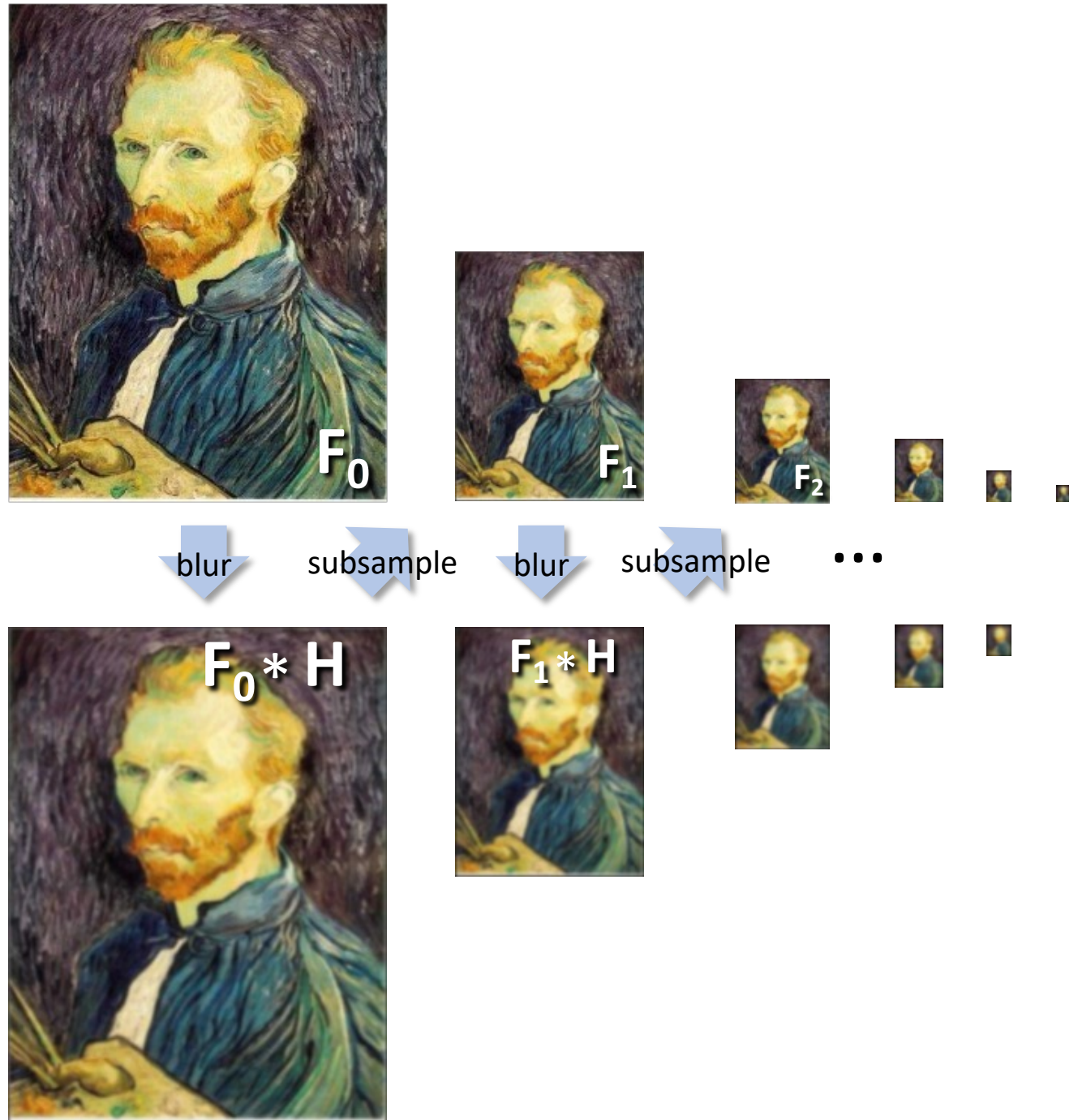
$$h[m, n] = \frac{1}{W_{mn}} \sum_{k, l} g[k, l] r_{mn}[k, l] f[m + k, n + l]$$

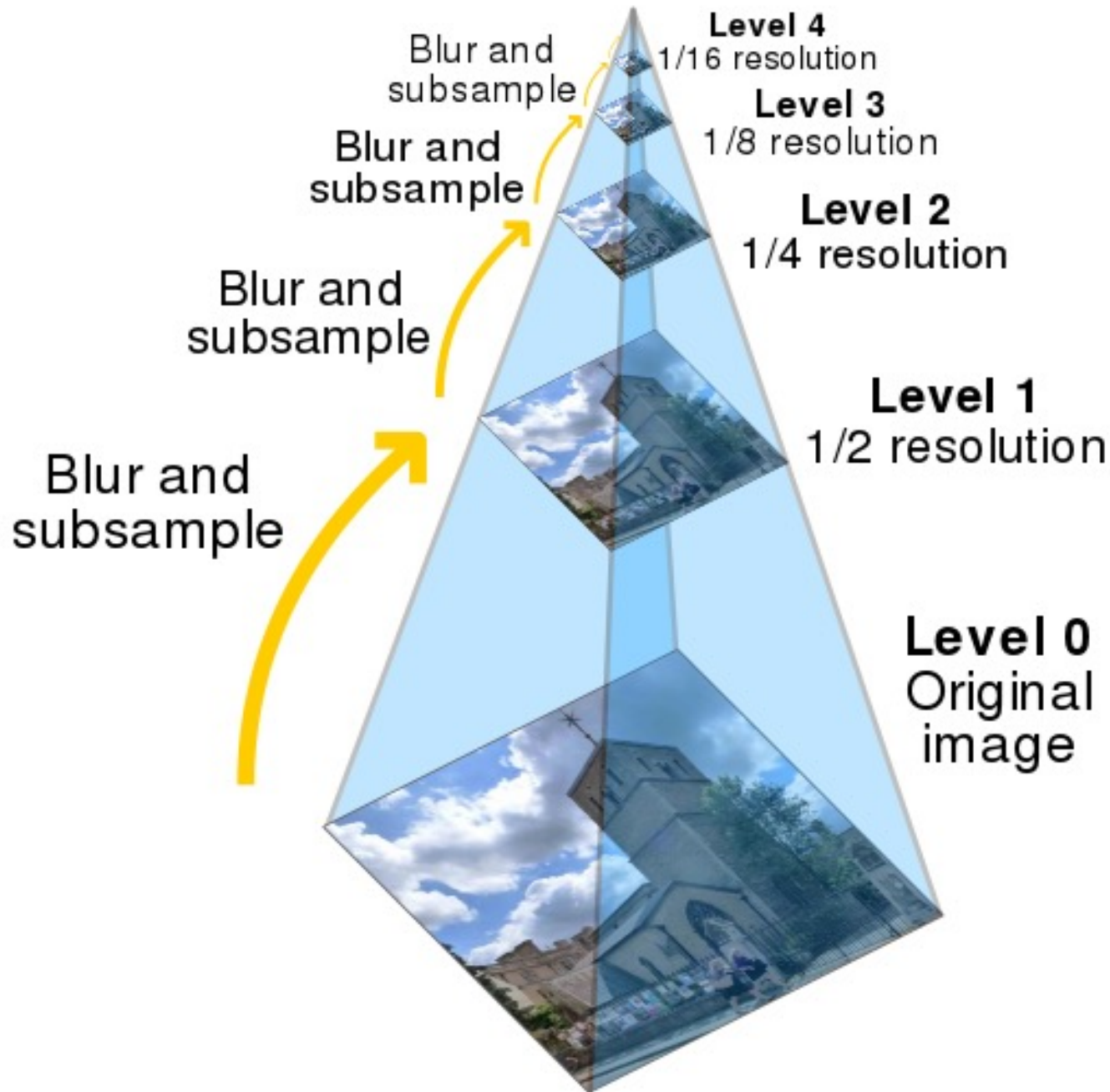
Aliasing

- 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 $\geq 2 \times$ 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.

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.

Today's Lecture

- Image Derivatives
- Edge Detection
- Fourier Analysis (in 1D)
- Fourier Analysis (in 2D)

Today's Lecture

- Image Derivatives
- Edge Detection
- Fourier Analysis (in 1D)
- Fourier Analysis (in 2D)

Partial derivatives with convolution

Image is function $f(x,y)$

Remember:
$$\frac{\partial f(x, y)}{\partial x} = \lim_{\epsilon \rightarrow 0} \frac{f(x + \epsilon, y) - f(x, y)}{\epsilon}$$

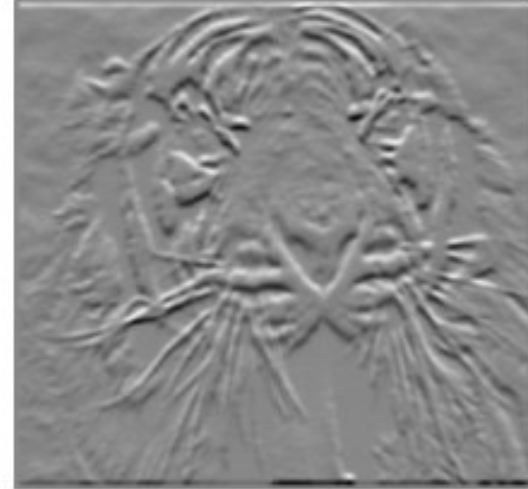
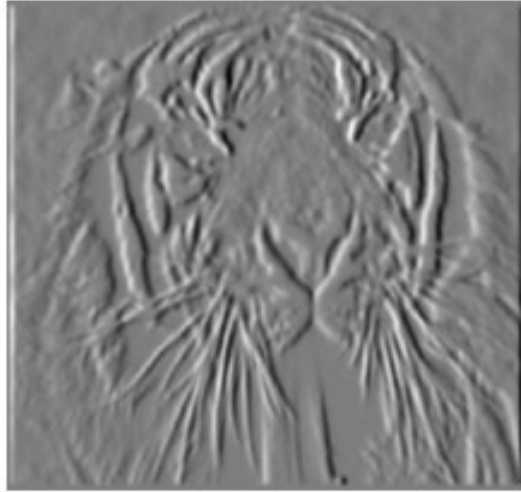
Approximate:
$$\frac{\partial f(x, y)}{\partial x} \approx \frac{f(x + 1, y) - f(x, y)}{1}$$

-1	1
----	---

Another one:
$$\frac{\partial f(x, y)}{\partial x} \approx \frac{f(x + 1, y) - f(x - 1, y)}{2}$$

-1	0	1
----	---	---

Image Gradient



-1	1
----	---

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

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

-1
1

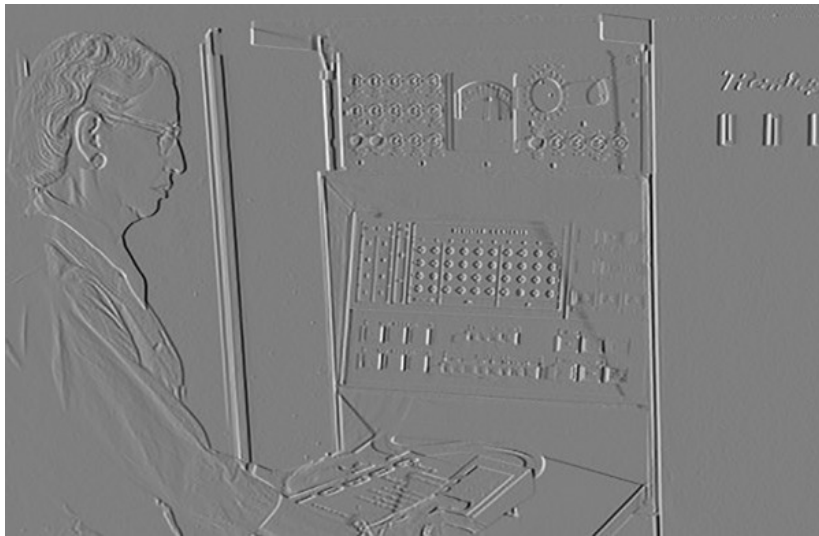
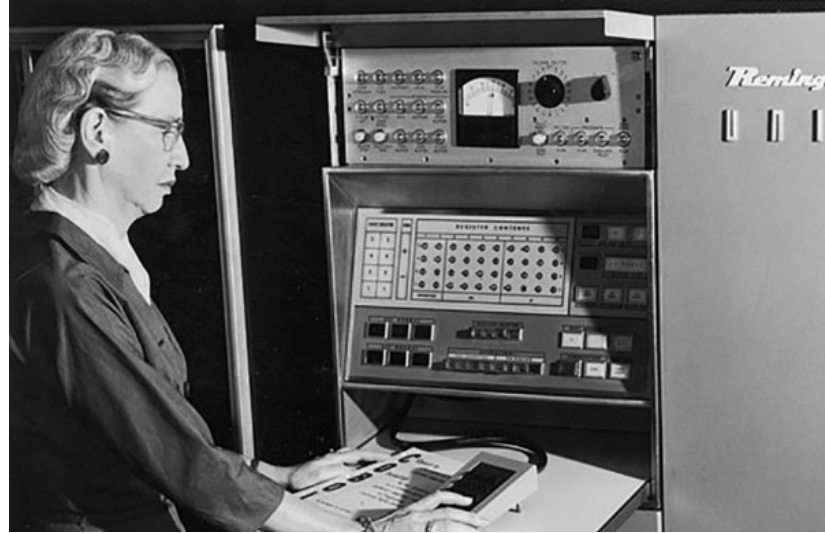
 or

1
-1

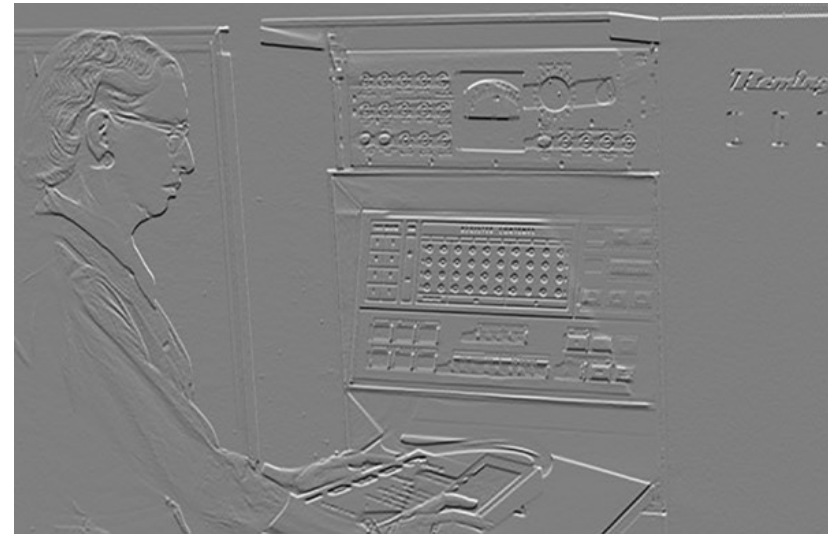
$$\|\nabla f\| = \sqrt{\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2}$$



Partial Derivatives



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



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

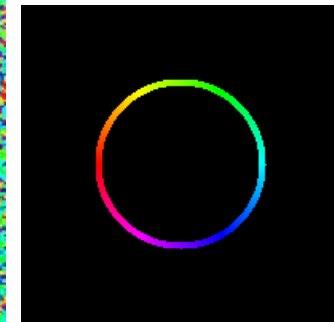
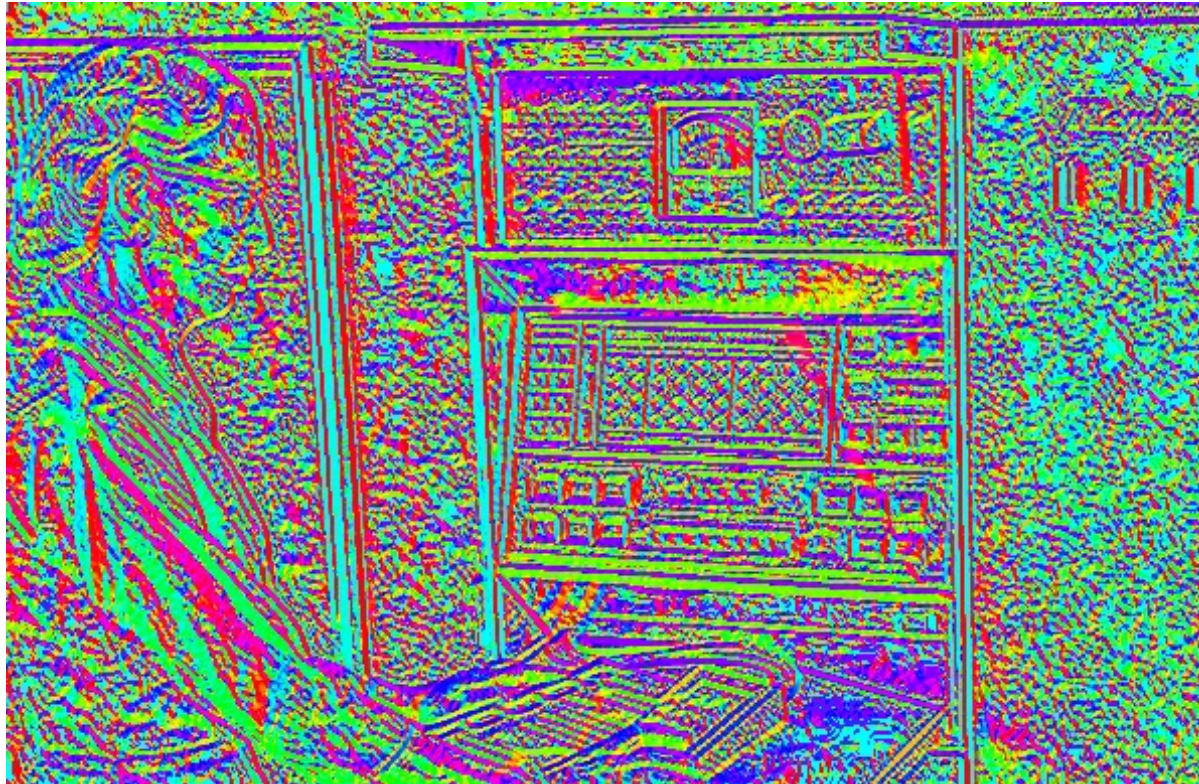
Gradient magnitude

$$\|\nabla f\| = \sqrt{\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2}$$



Gradient Orientation

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

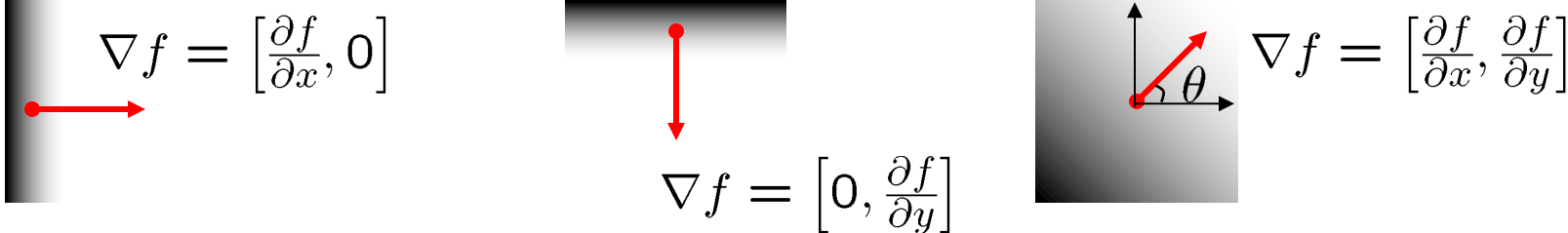


all the gradients

Image gradient

- The gradient of an image:

$$\nabla f = \left[\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y} \right]$$

-  $\nabla f = \left[\frac{\partial f}{\partial x}, 0 \right]$ $\nabla f = \left[0, \frac{\partial f}{\partial y} \right]$ $\nabla f = \left[\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y} \right]$

The gradient points in the direction of most rapid increase in intensity

- How does this direction relate to the direction of the edge?

The *edge strength* is given by the gradient magnitude

$$\|\nabla f\| = \sqrt{\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2}$$

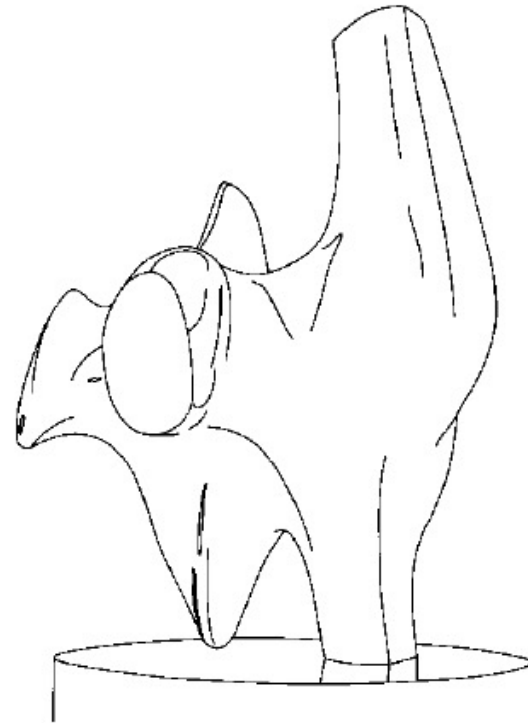
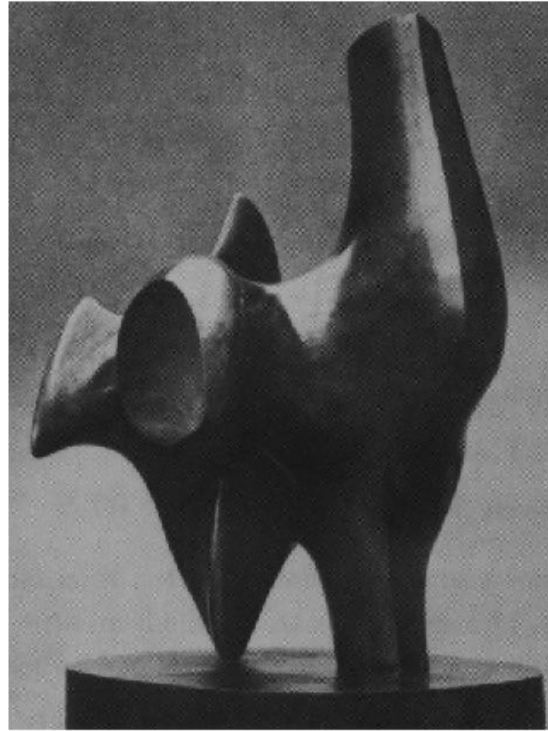
The gradient direction is given by $\theta = \tan^{-1} \left(\frac{\partial f}{\partial y} / \frac{\partial f}{\partial x} \right)$

Quiz time ...

Today's Lecture

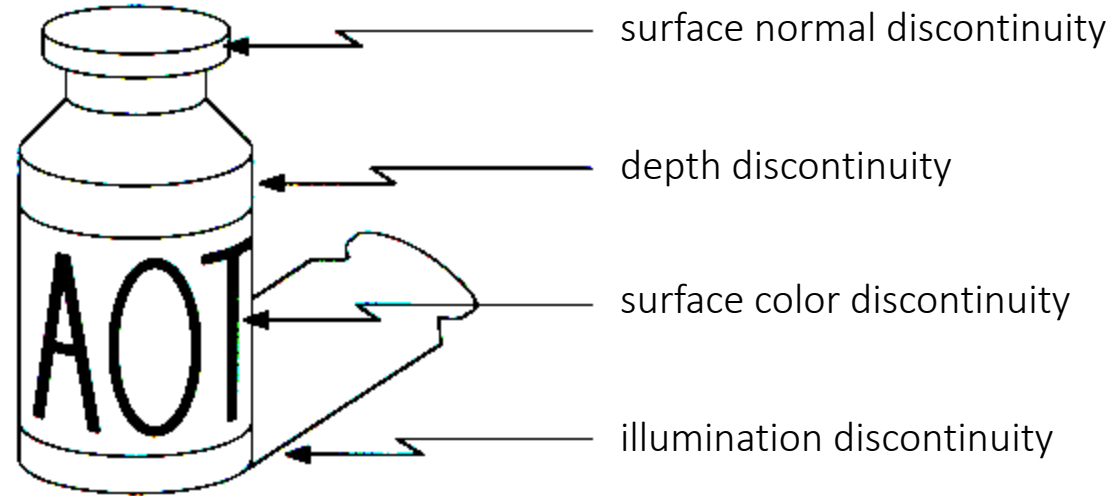
- Image Derivatives
- **Edge Detection**
- Fourier Analysis (in 1D)
- Fourier Analysis (in 2D)

Edge detection



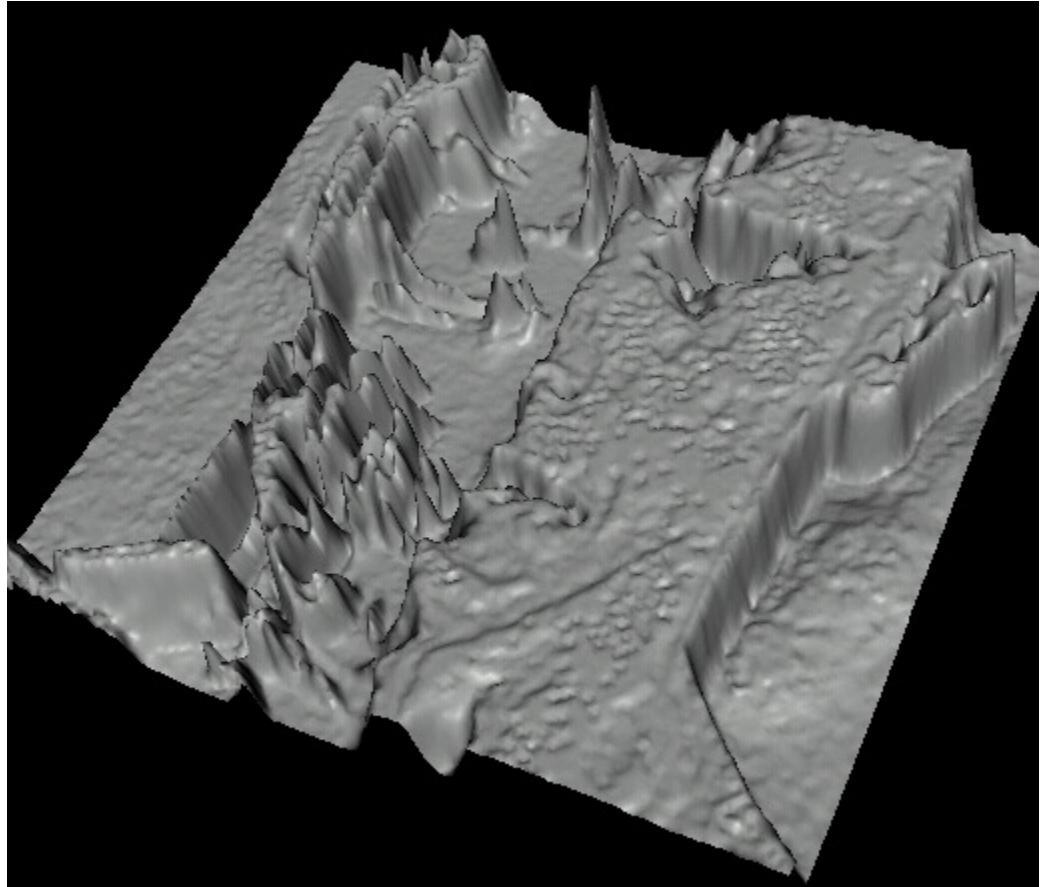
- Convert a 2D image into a set of curves
 - Extracts salient features of the scene
 - More compact than pixels

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

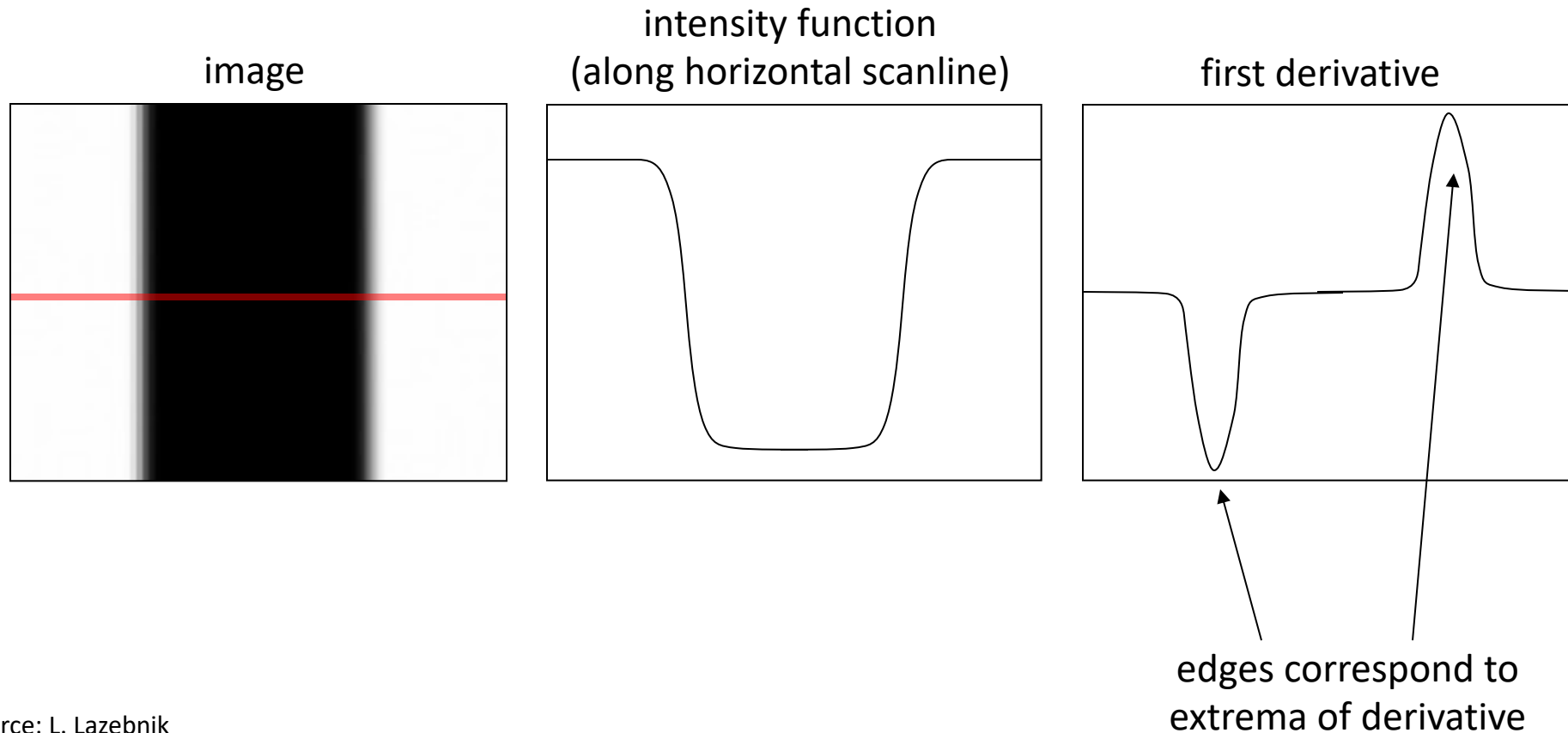
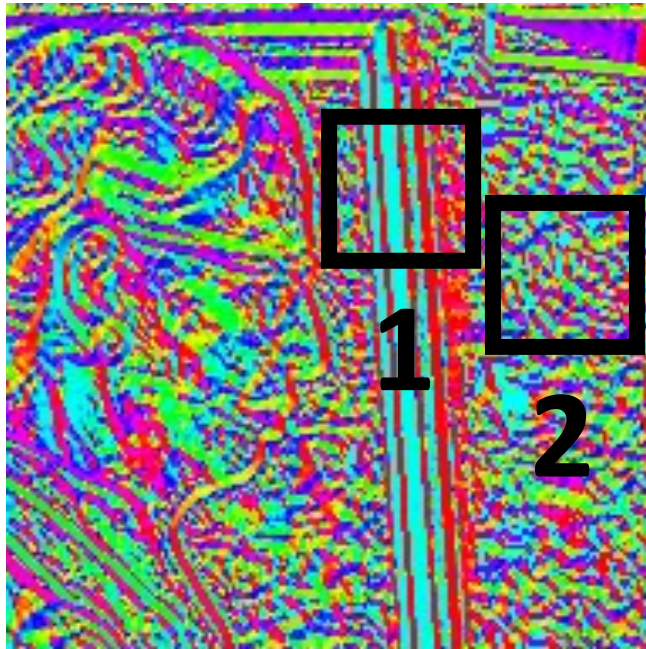


Image Gradient & Edges

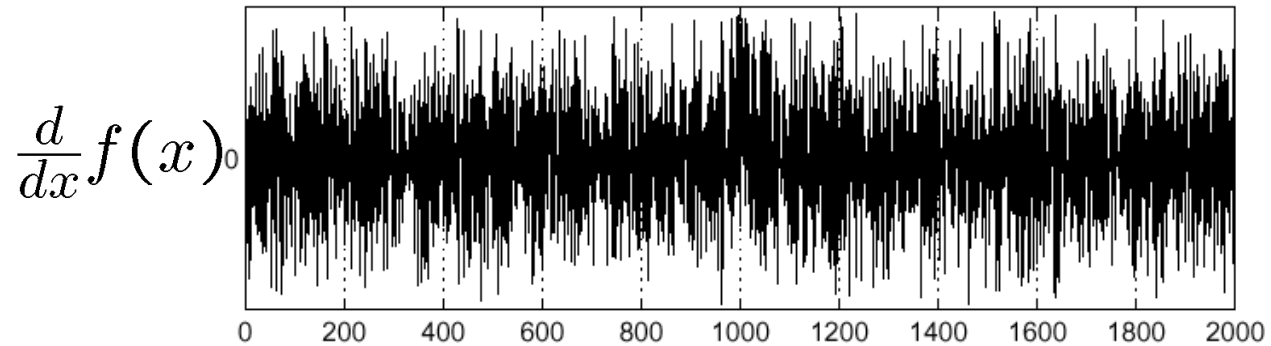
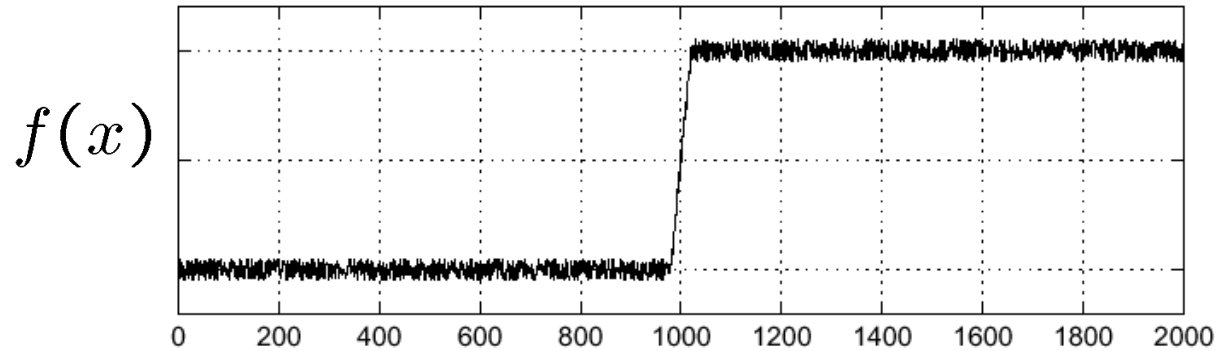
$$\theta = \tan^{-1} \left(\frac{\partial f}{\partial y} / \frac{\partial f}{\partial x} \right) \quad \text{Direction of image gradients}$$

Why is there structure at 1 and not at 2?



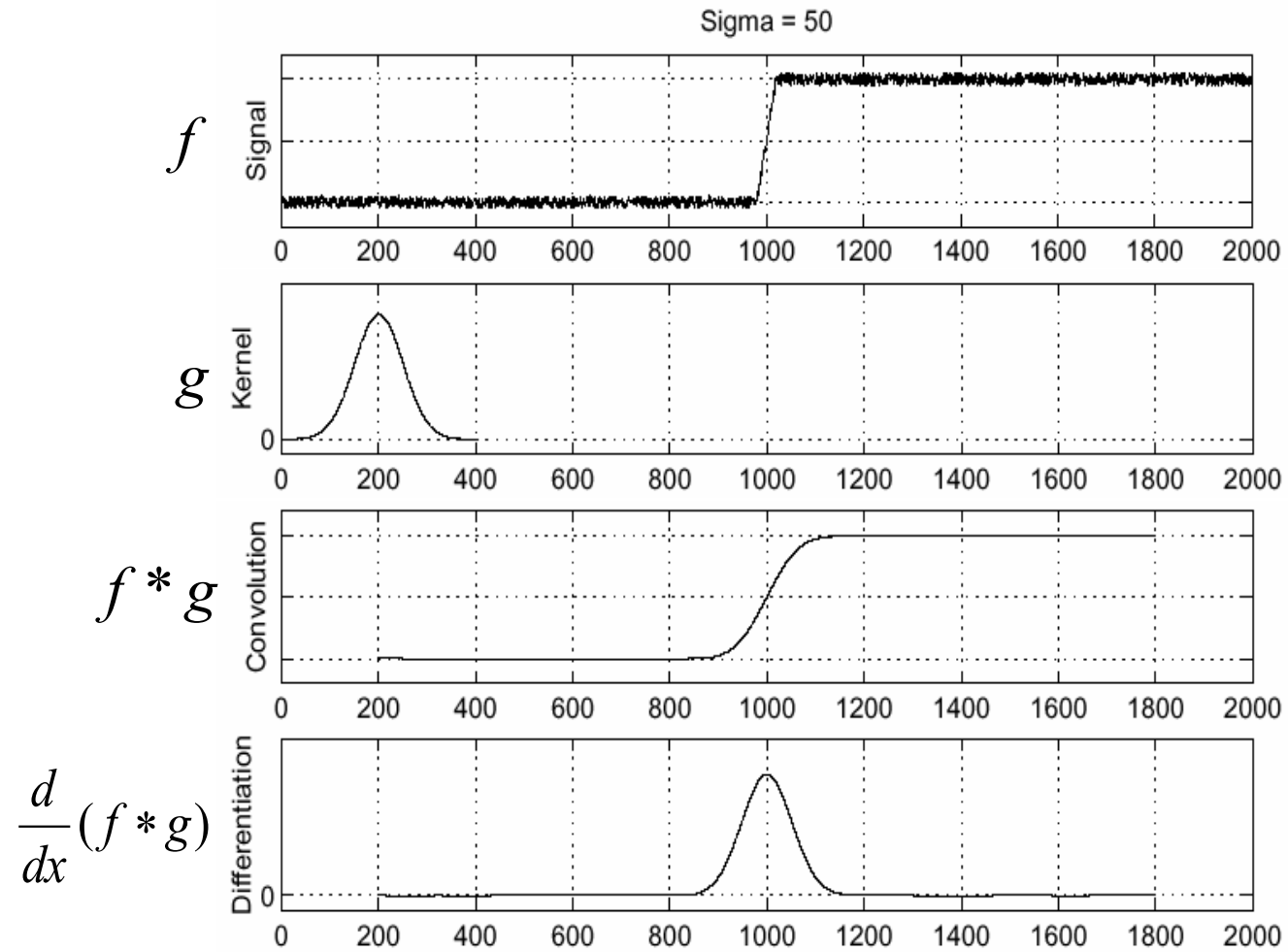
Effects of noise

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



Where is the edge?

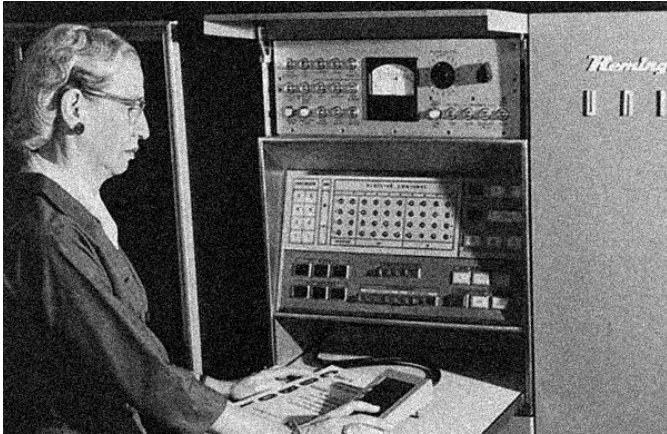
Solution: smooth first



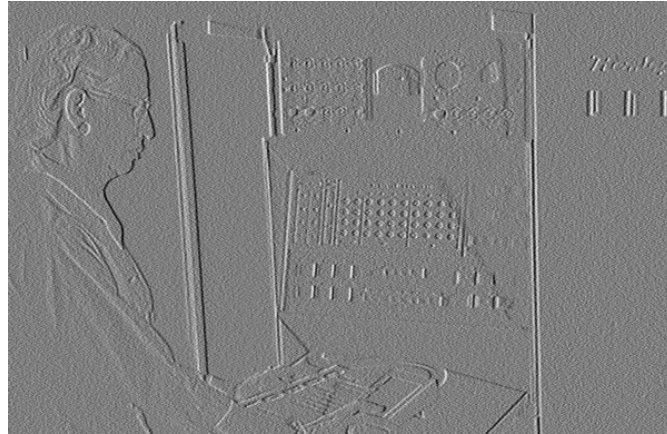
- To find edges, look for peaks in $\frac{d}{dx}(f * g)$

Noise in 2D

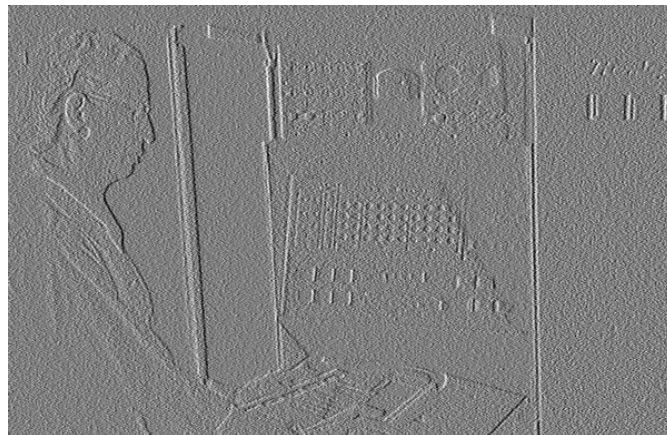
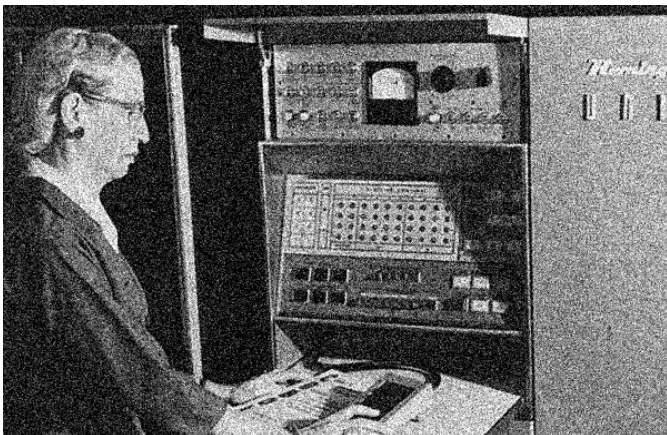
Noisy Input



I_x via $[-1,0,1]$



Zoom

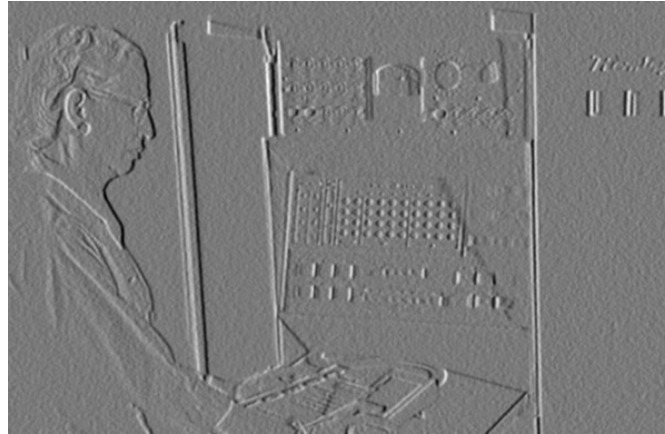


Noise + Smoothing

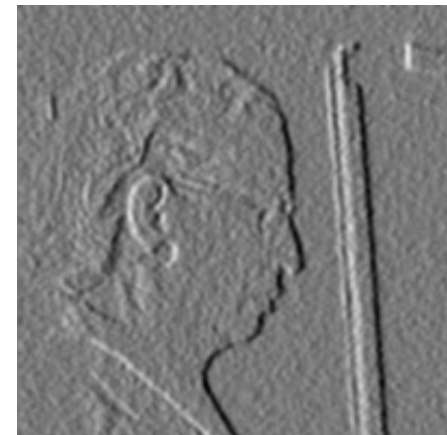
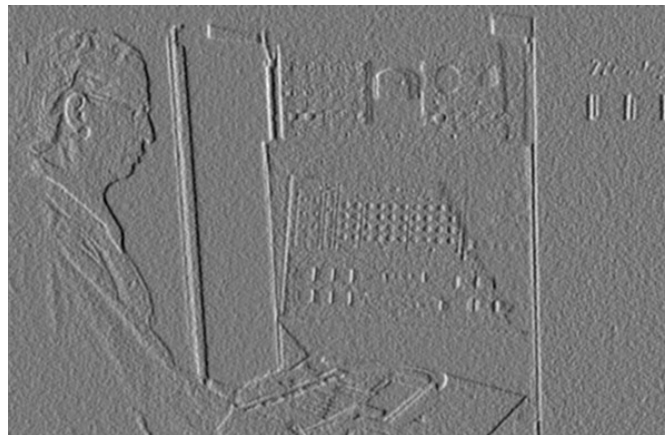
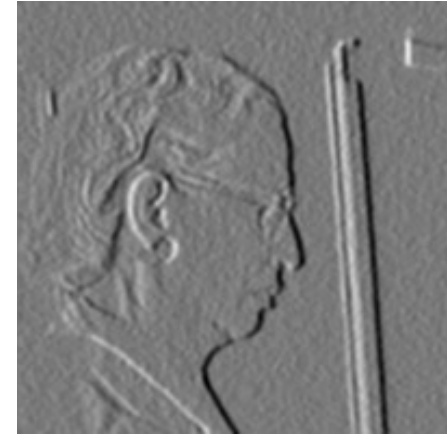
Smoothed Input



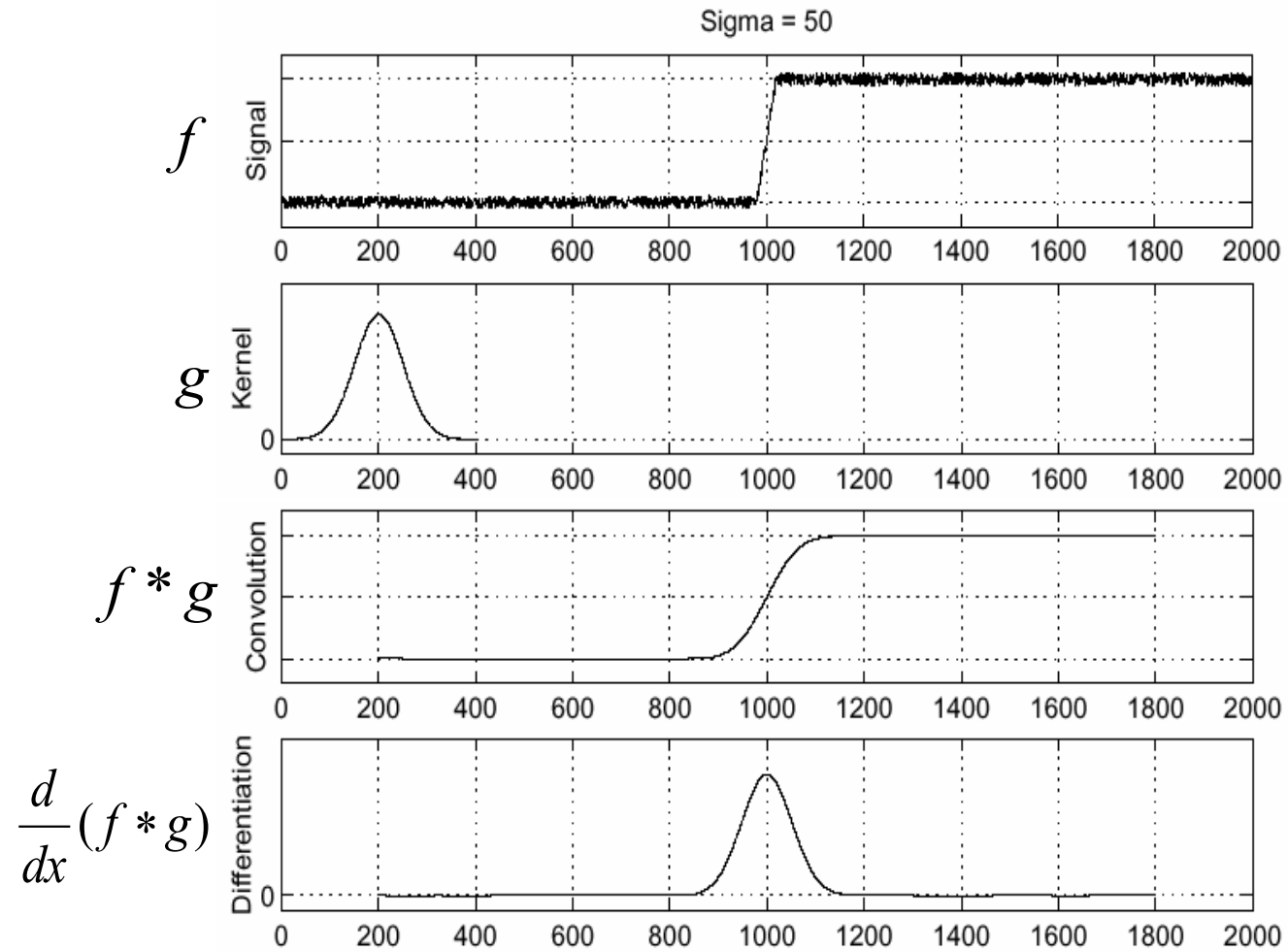
I_x via $[-1,0,1]$



Zoom



How many convolutions here?



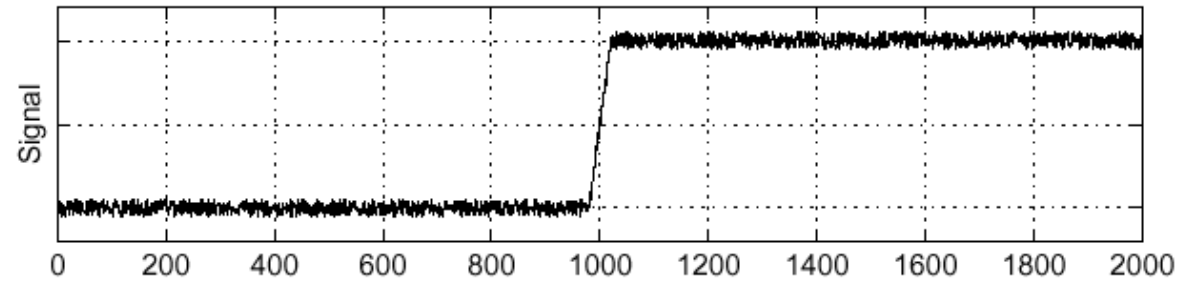
can we reduce this?

Derivative theorem of convolution

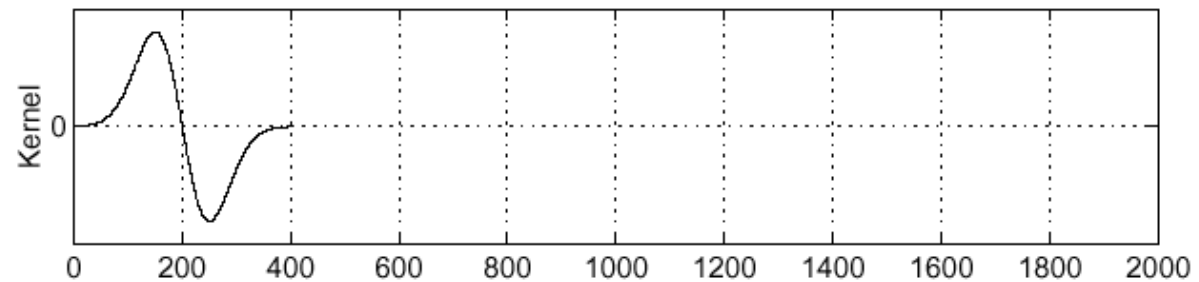
$$\frac{\partial}{\partial x}(h \star f) = \left(\frac{\partial}{\partial x}h\right) \star f$$

- This saves us one operation:

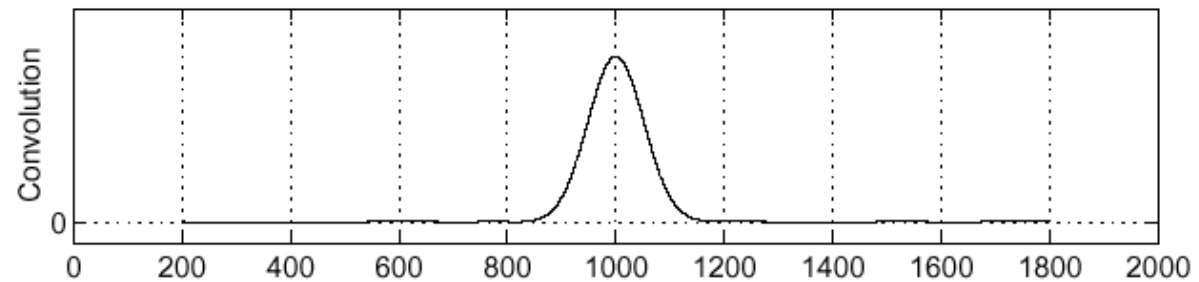
Sigma = 50



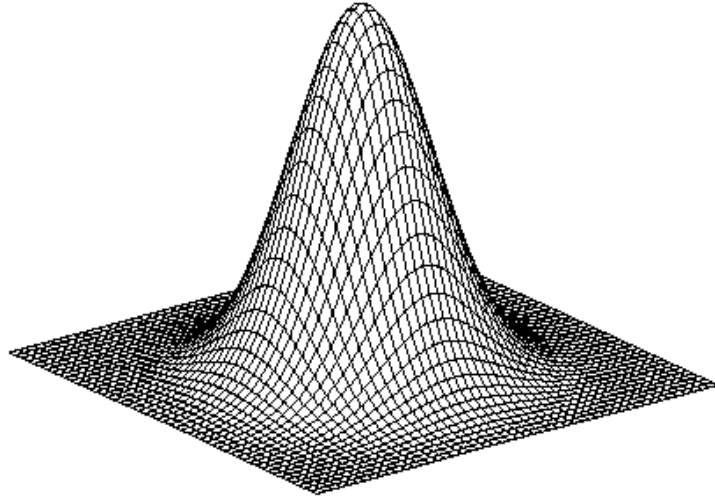
$$\frac{\partial}{\partial x}h$$



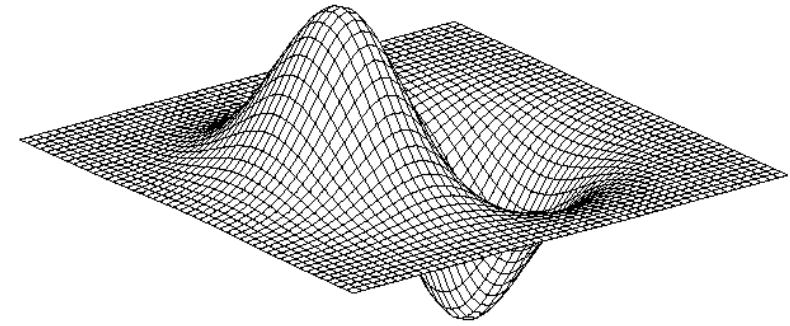
$$\left(\frac{\partial}{\partial x}h\right) \star f$$



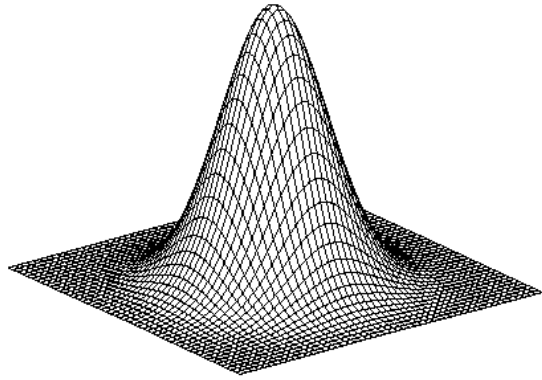
Derivative of Gaussian filter



$$* [1 \ -1] =$$

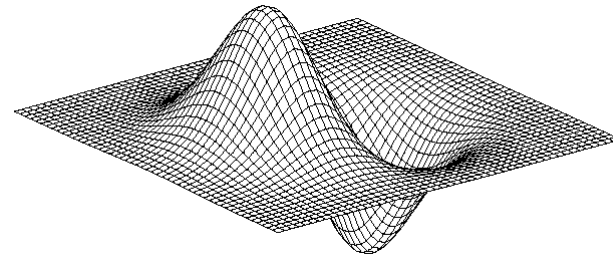


2D edge detection filters



Gaussian

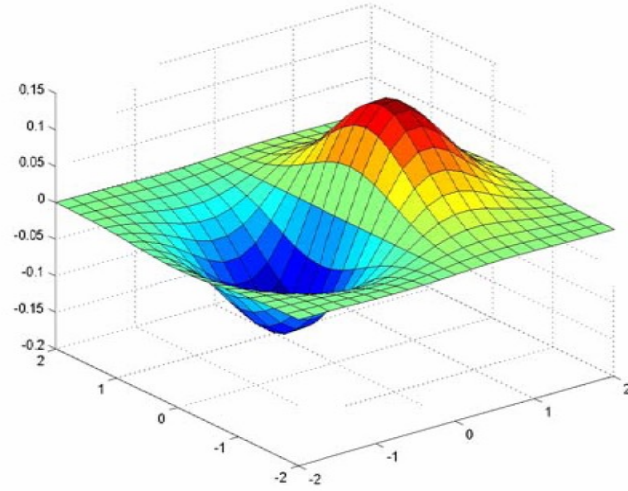
$$h_{\sigma}(u, v) = \frac{1}{2\pi\sigma^2} e^{-\frac{u^2+v^2}{2\sigma^2}}$$



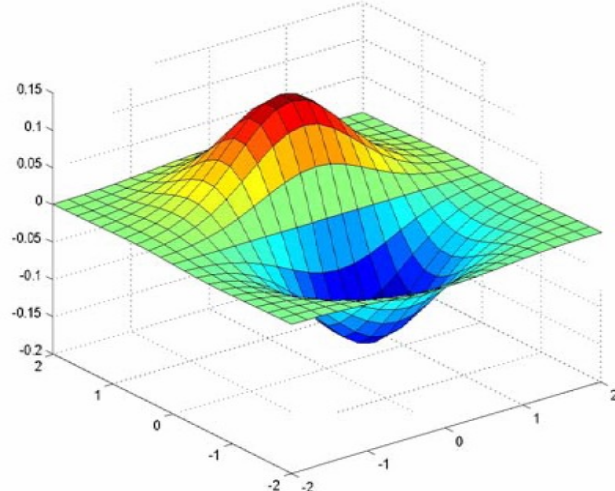
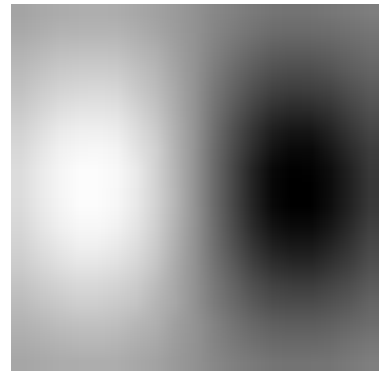
derivative of Gaussian (x)

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

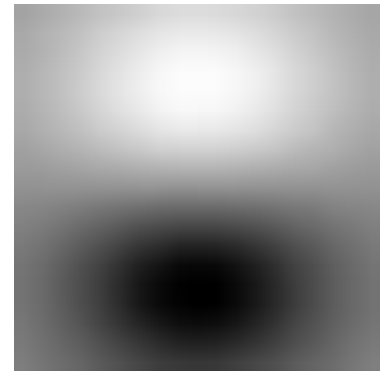
Derivative of Gaussian filter



x-direction



y-direction



The Sobel operator

- Common approximation of derivative of Gaussian

$$\frac{1}{8} \begin{array}{|c|c|c|} \hline -1 & 0 & 1 \\ \hline -2 & 0 & 2 \\ \hline -1 & 0 & 1 \\ \hline \end{array}$$

s_x

$$\frac{1}{8} \begin{array}{|c|c|c|} \hline 1 & 2 & 1 \\ \hline 0 & 0 & 0 \\ \hline -1 & -2 & -1 \\ \hline \end{array}$$

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

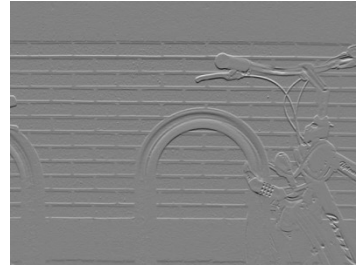
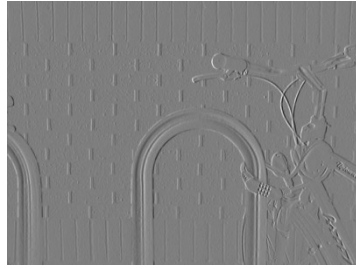
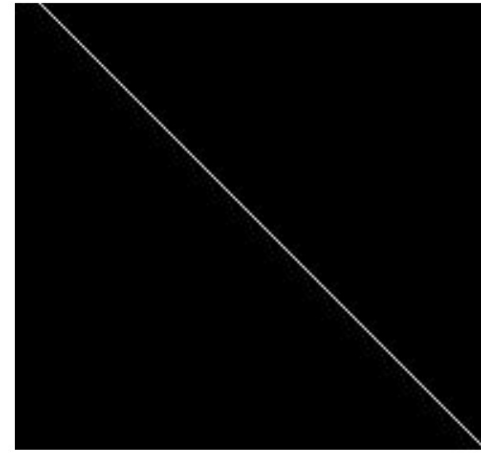




Image with Edge



Edge Location

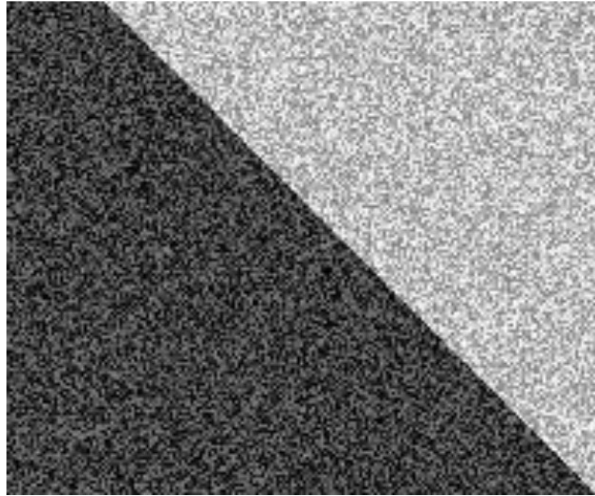
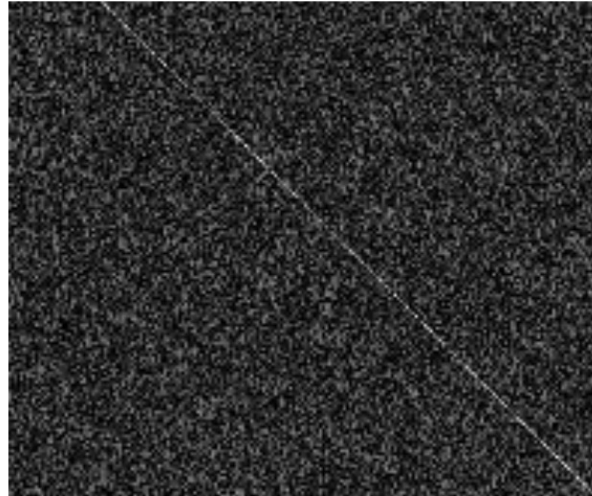
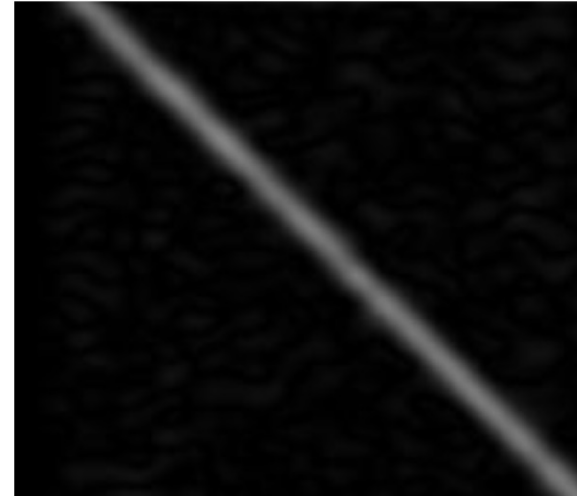


Image + Noise



Derivatives detect edge *and* noise



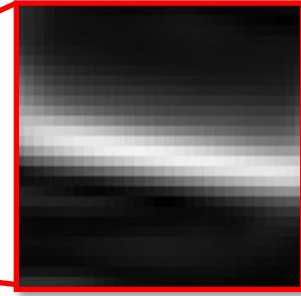
Smoothed derivative removes noise, but blurs edge

Example



original image

Finding edges

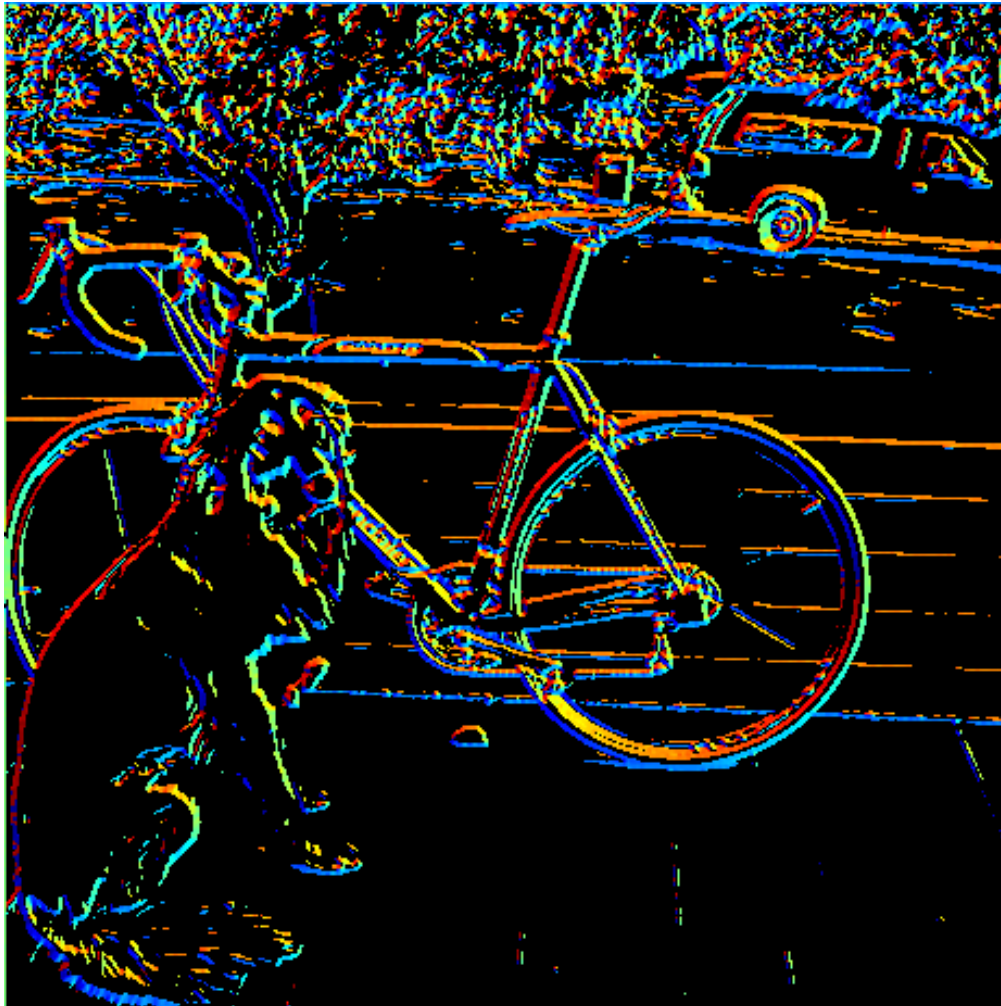


where is the edge?

smoothed gradient magnitude

Get Orientation at Each Pixel

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



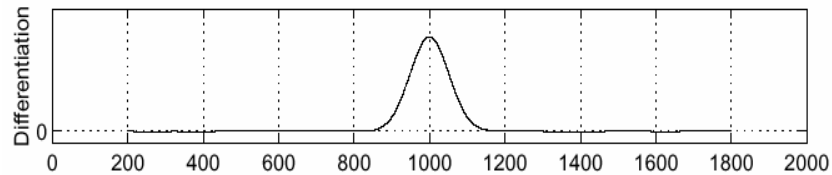
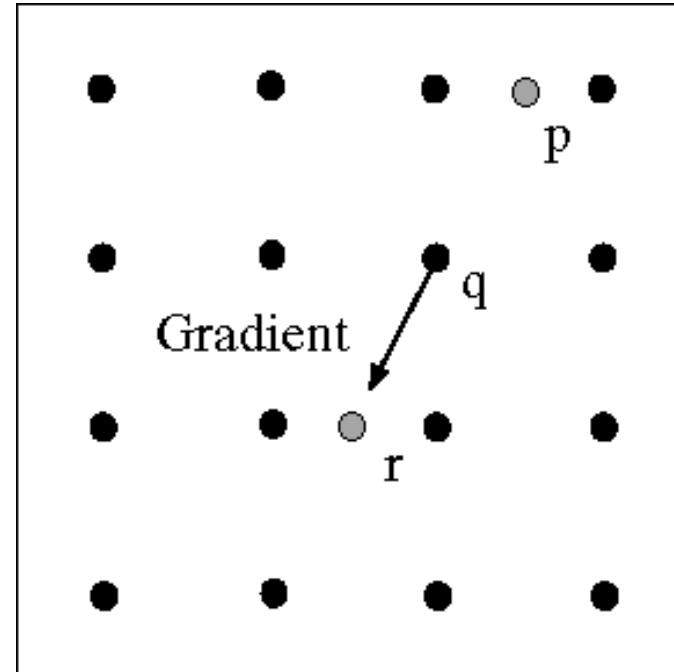
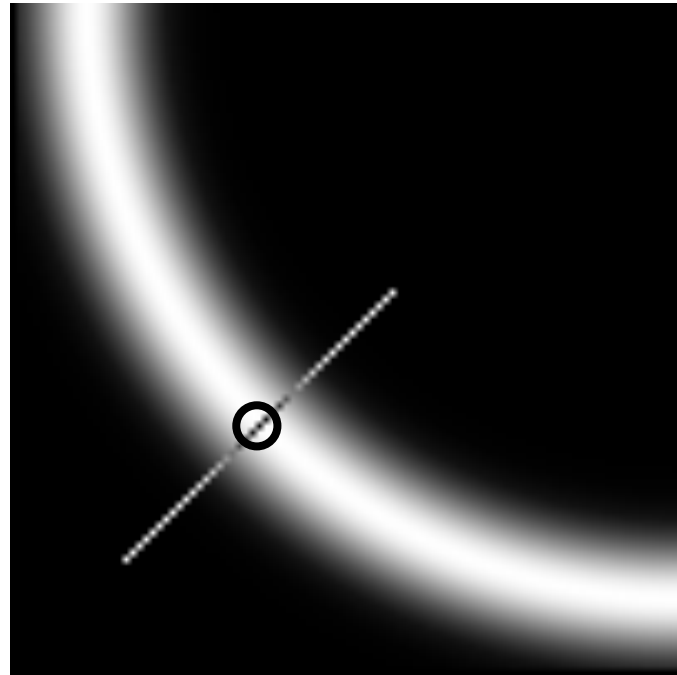
$$\text{theta} = \text{atan2}(g_y, g_x)$$

360

Gradient orientation angle

0

Non-maximum suppression

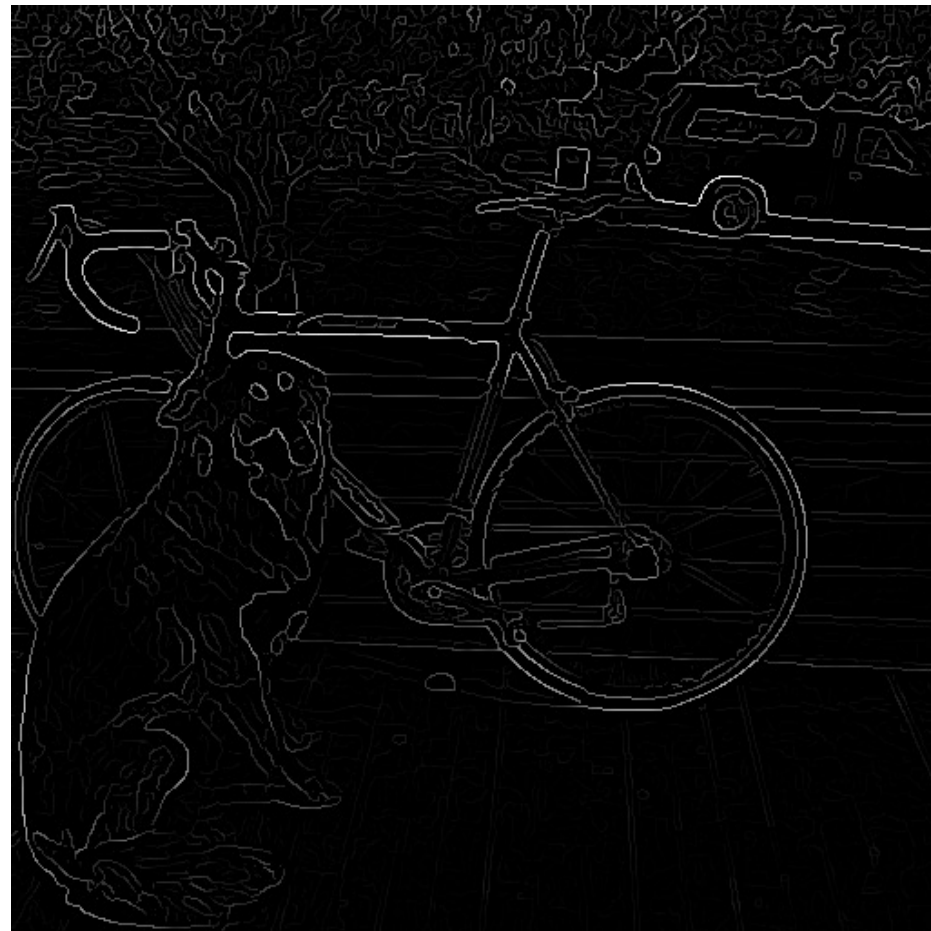


- Check if pixel is local maximum along gradient direction
 - requires *interpolating* pixels p and r

Before Non-max Suppression



After Non-max Suppression



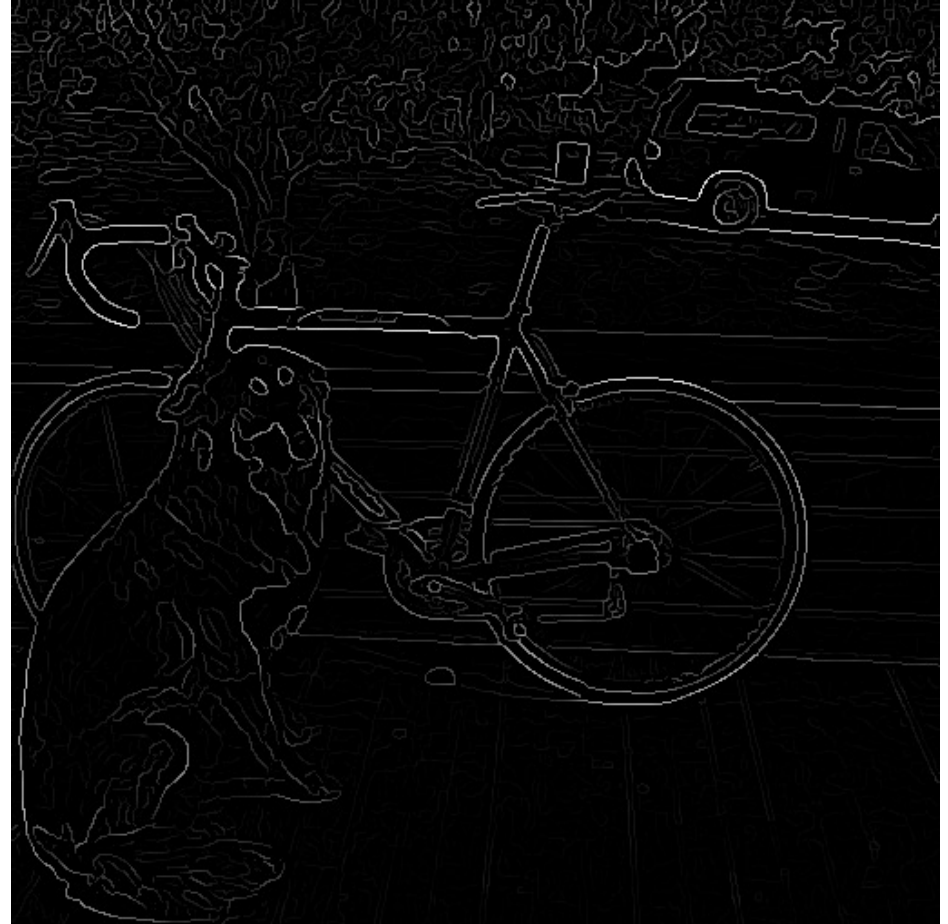
Still noise exists!



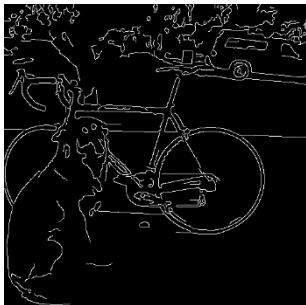
Thresholding edges

- 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

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

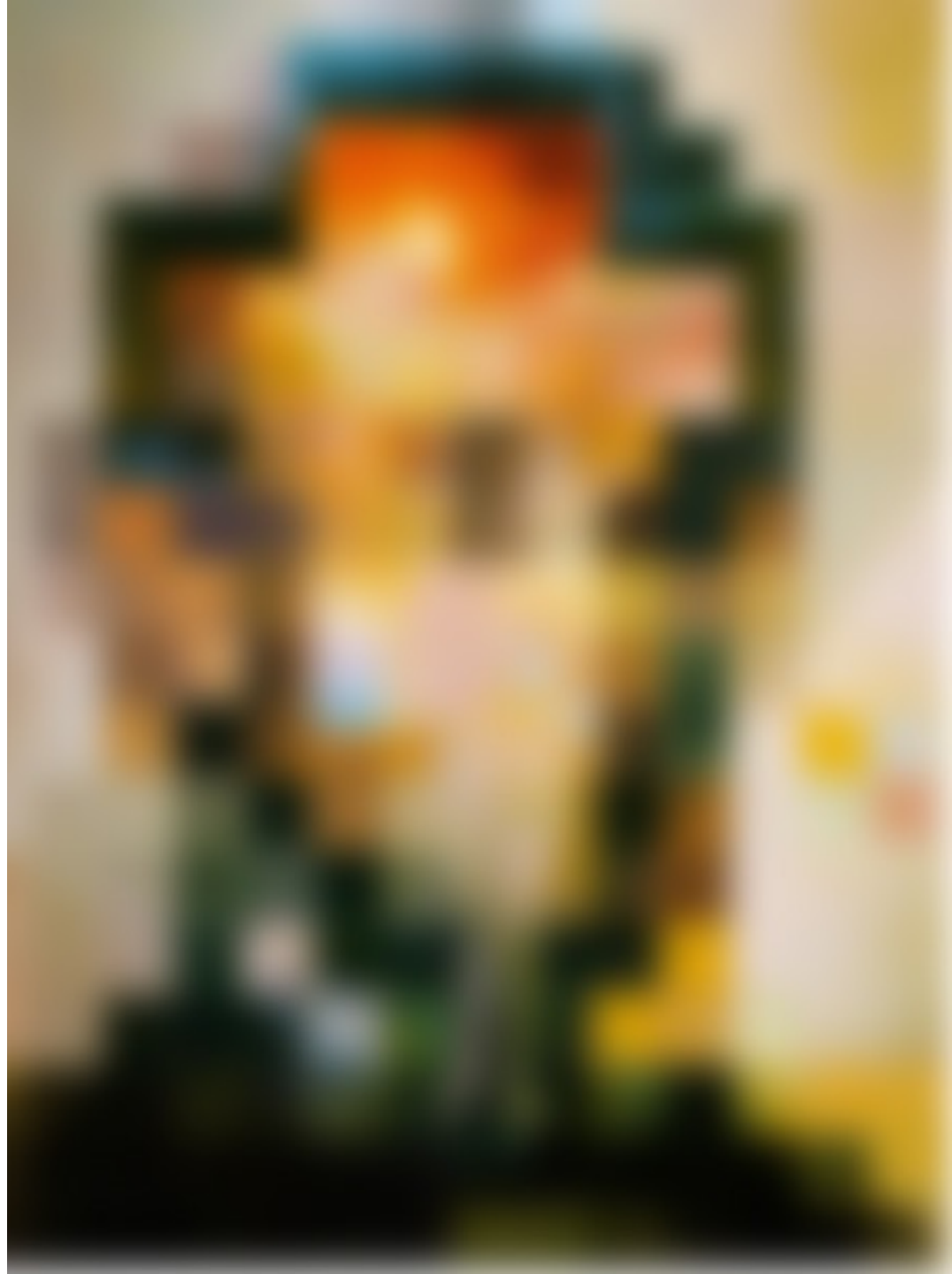
Today's Lecture

- Image Derivatives
- Edge Detection
- **Fourier Analysis (in 1D)**
- Fourier Analysis (in 2D)



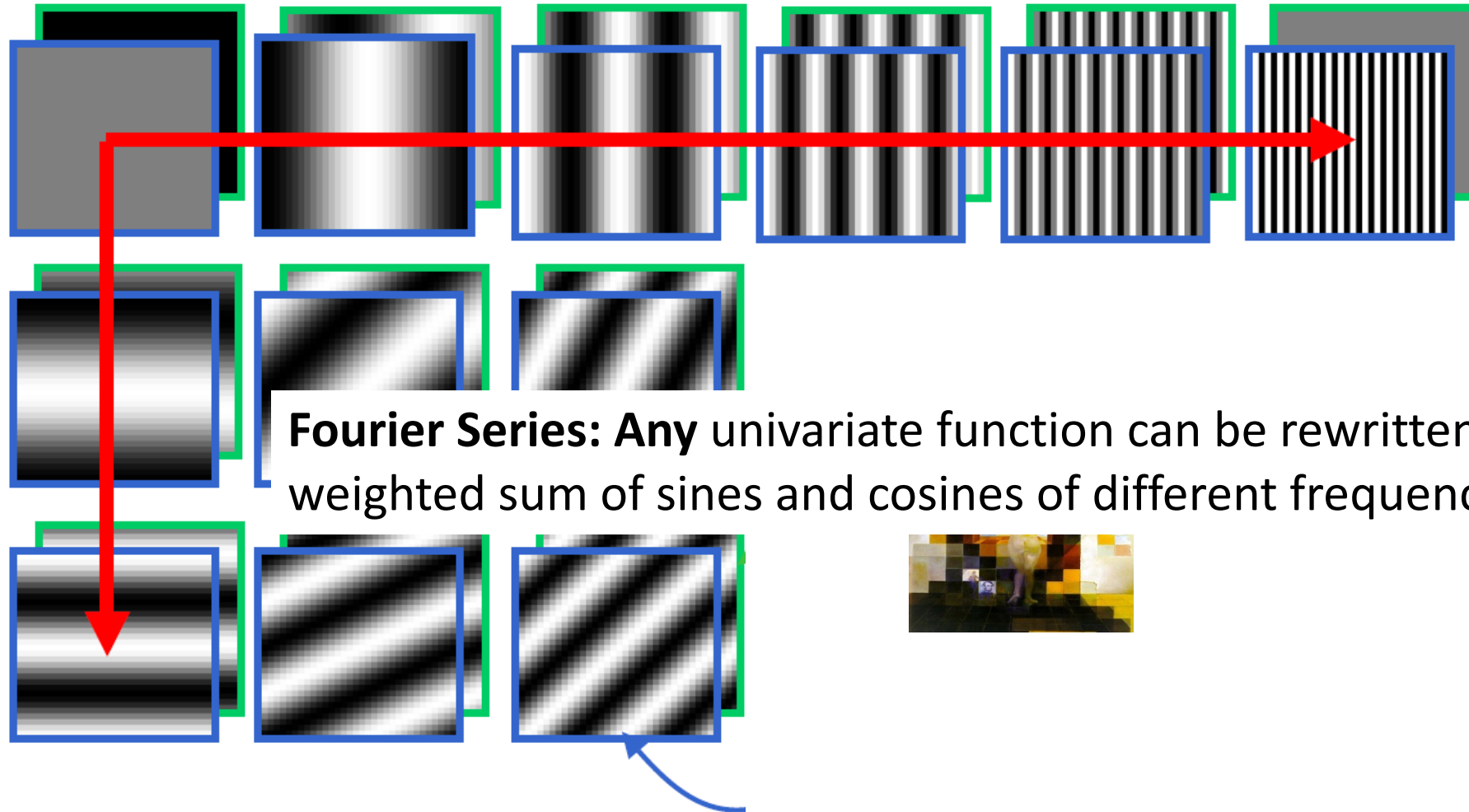
Salvador Dalí

*"Gala Contemplating the Mediterranean Sea,
which at 30 meters becomes the portrait
of Abraham Lincoln", 1976*



A nice set of basis

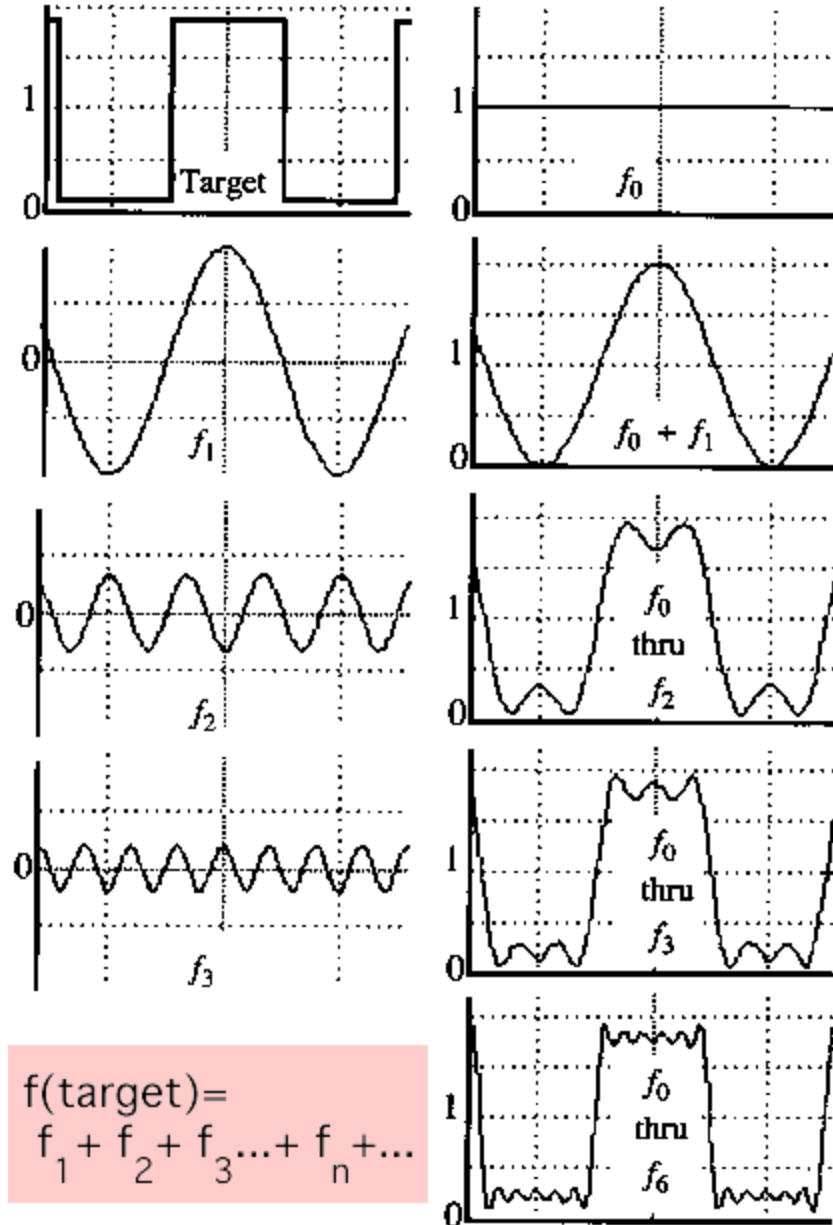
Teases away fast vs. slow changes in the image.



This change of basis has a special name...

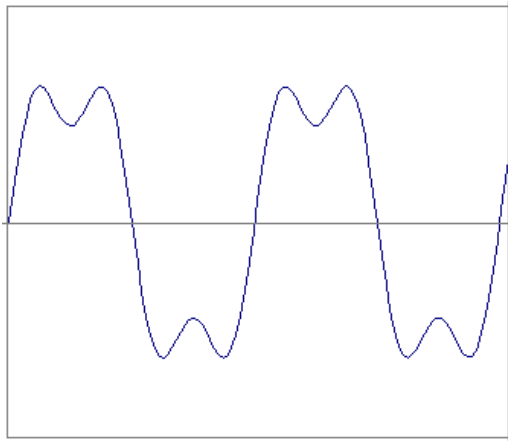
A sum of sines

- Our building block:
- $A \sin(\omega x + \phi)$
- Add enough of them to get any signal $f(x)$ you want!
- A = Amplitude (strength/intensity)
- ω = frequency (how fast or slow)
- ϕ = phase (starting position of the sinusoid)



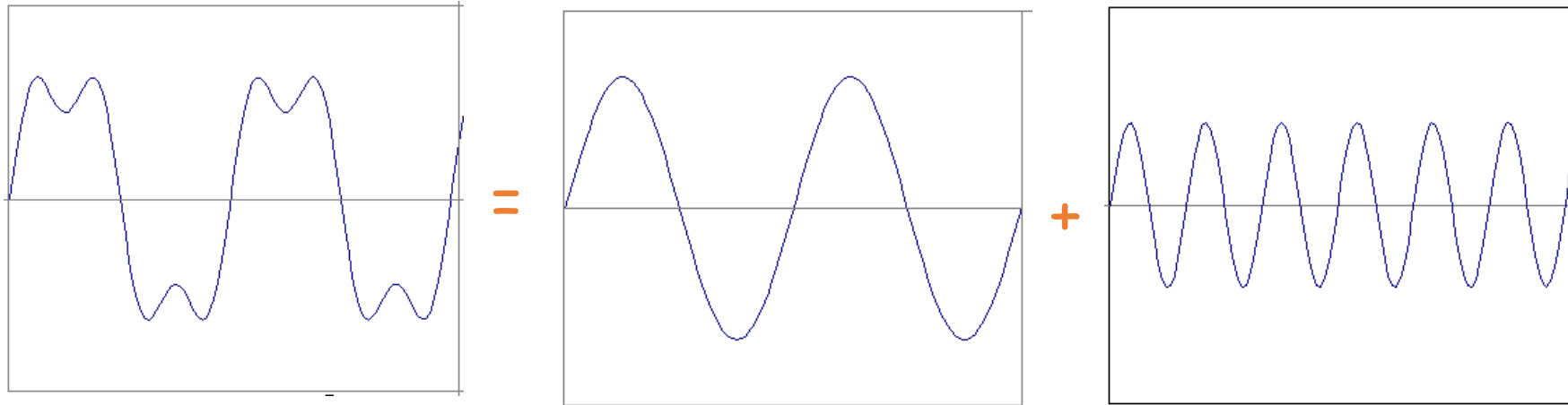
Time and Frequency

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



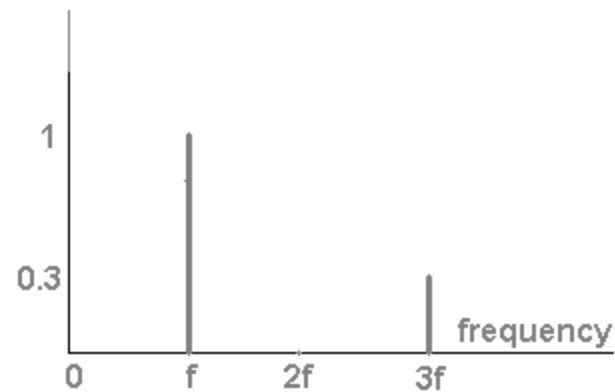
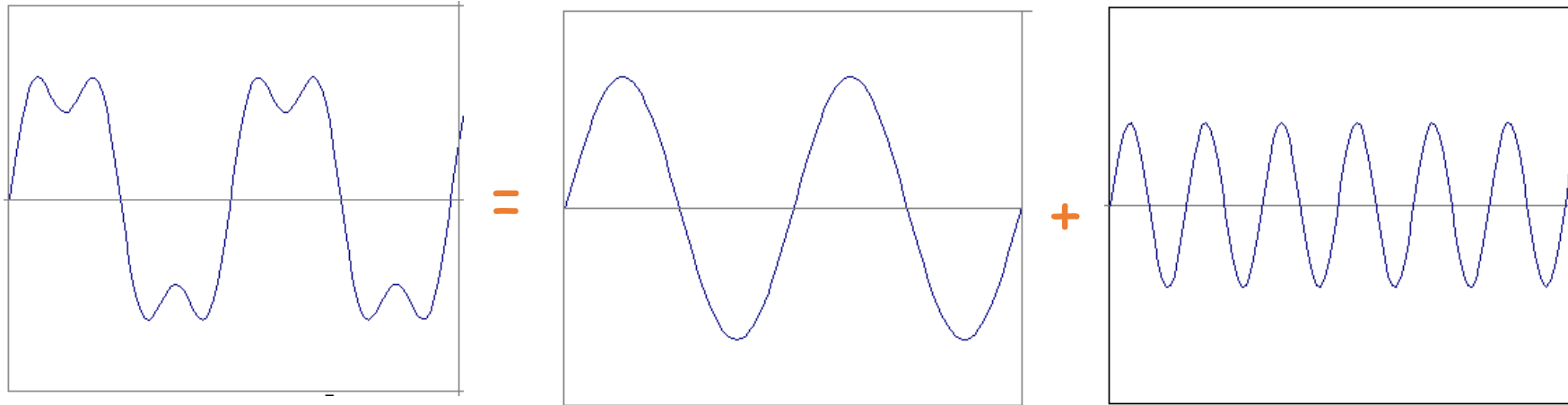
Time and Frequency

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



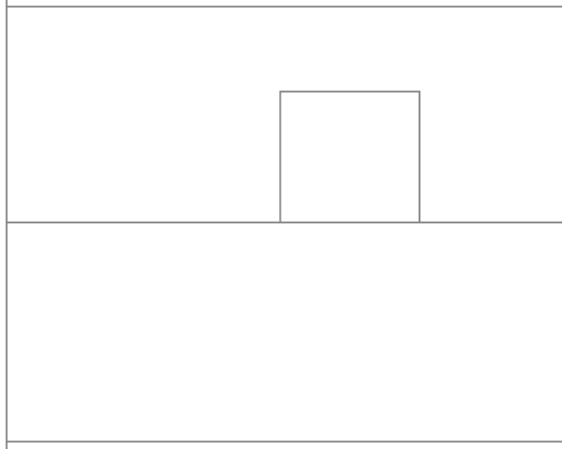
Frequency Spectra

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

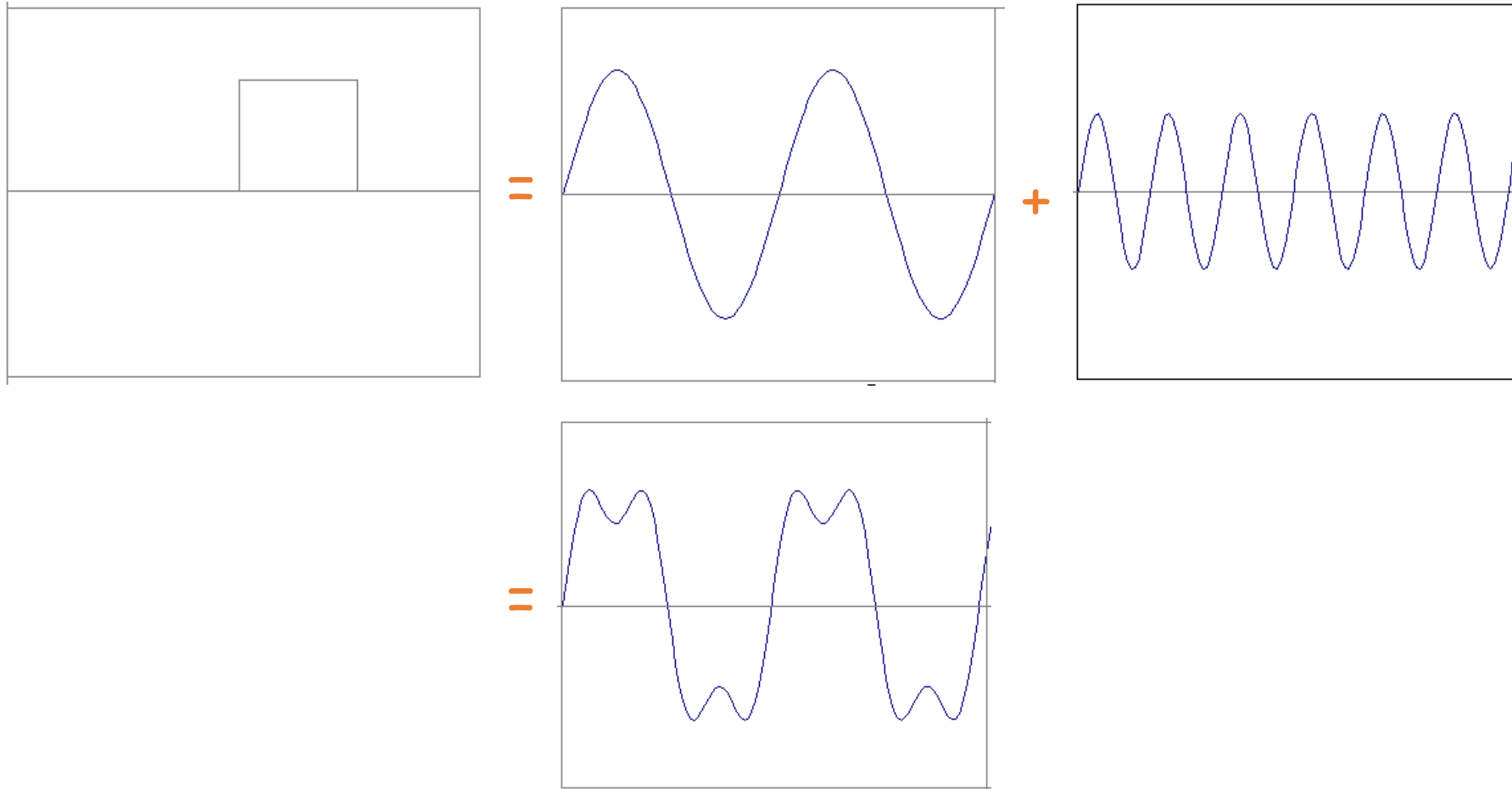


Frequency Spectra

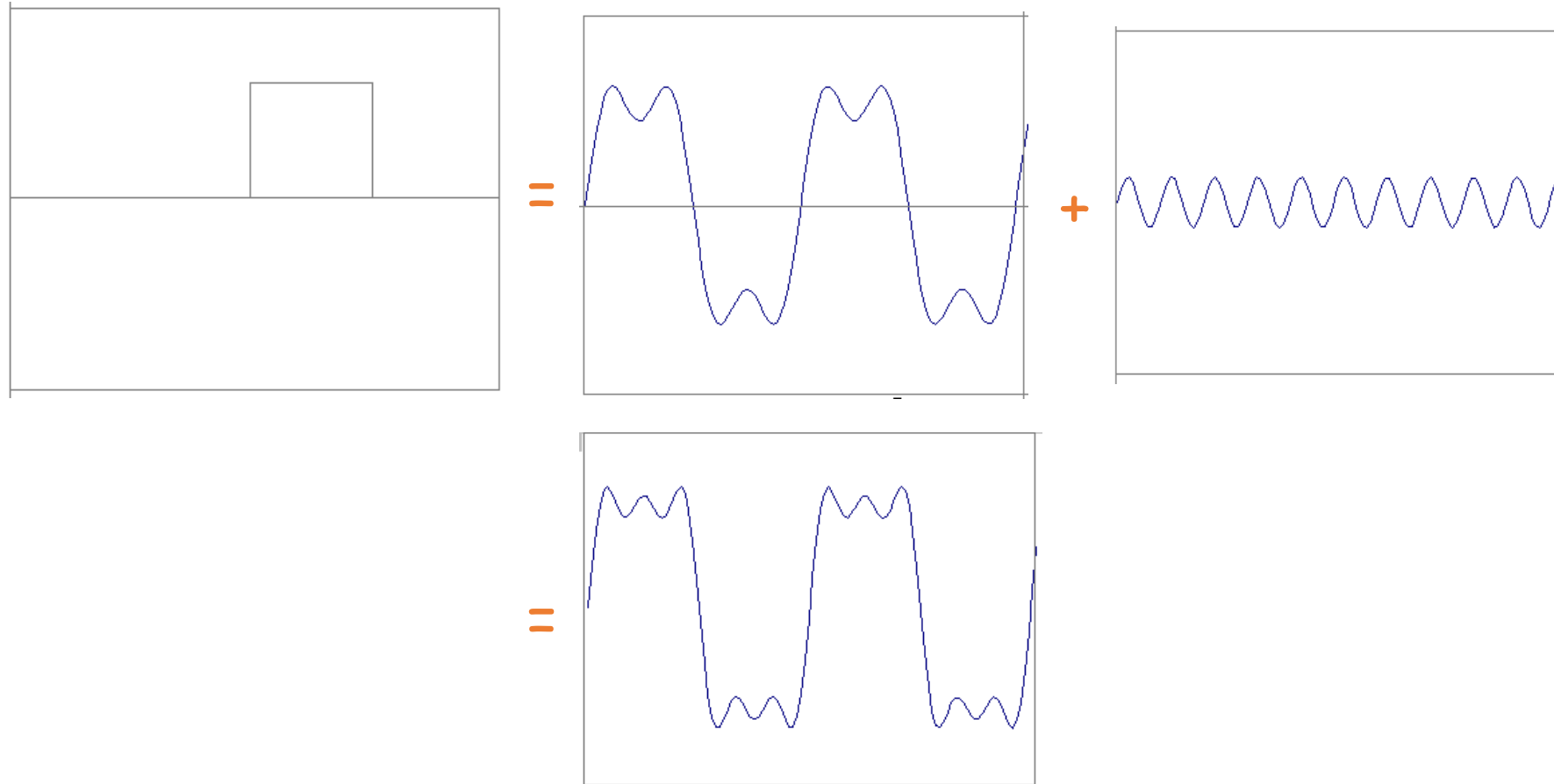
- Usually, frequency is more interesting than the phase



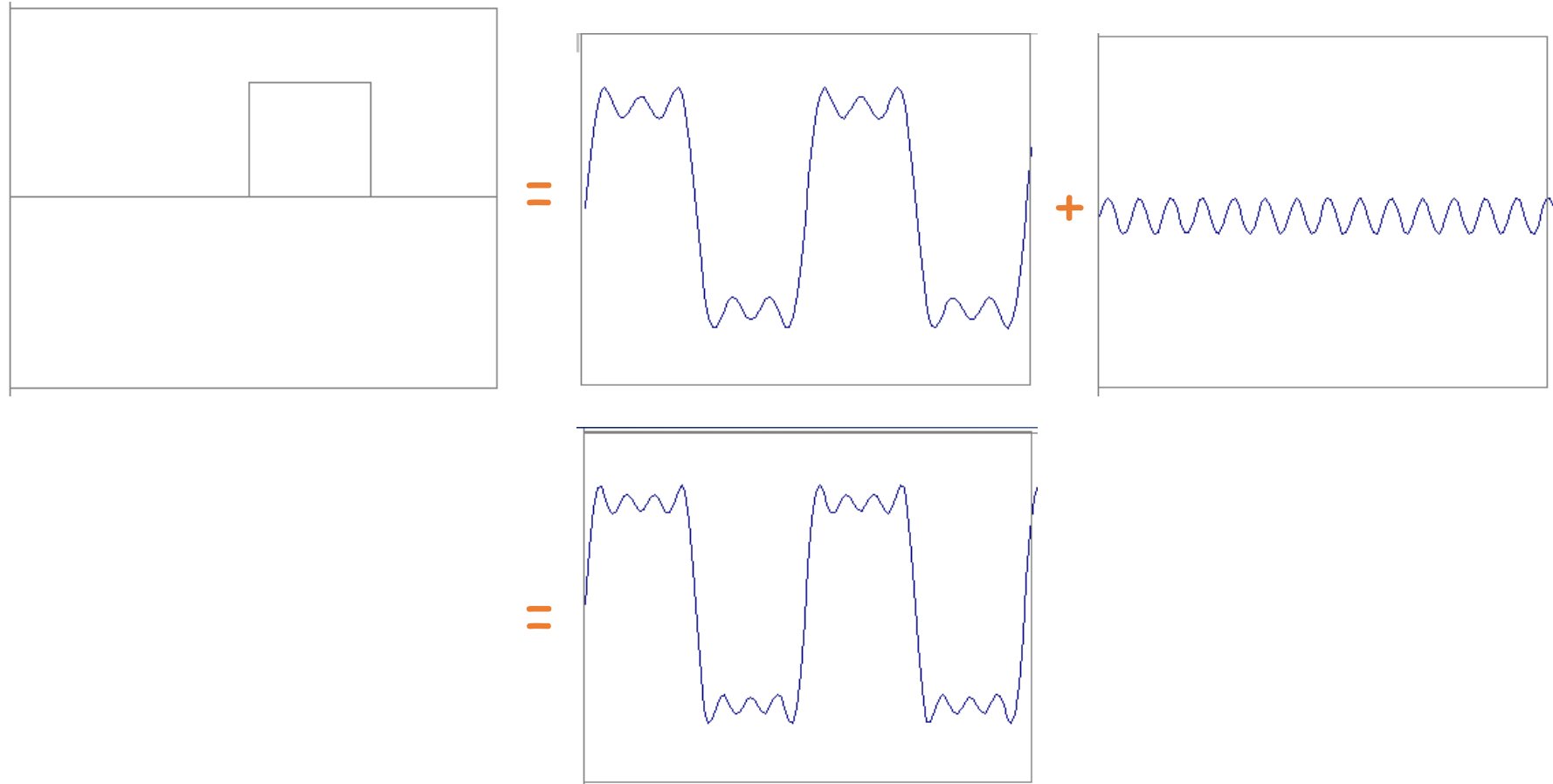
Frequency Spectra



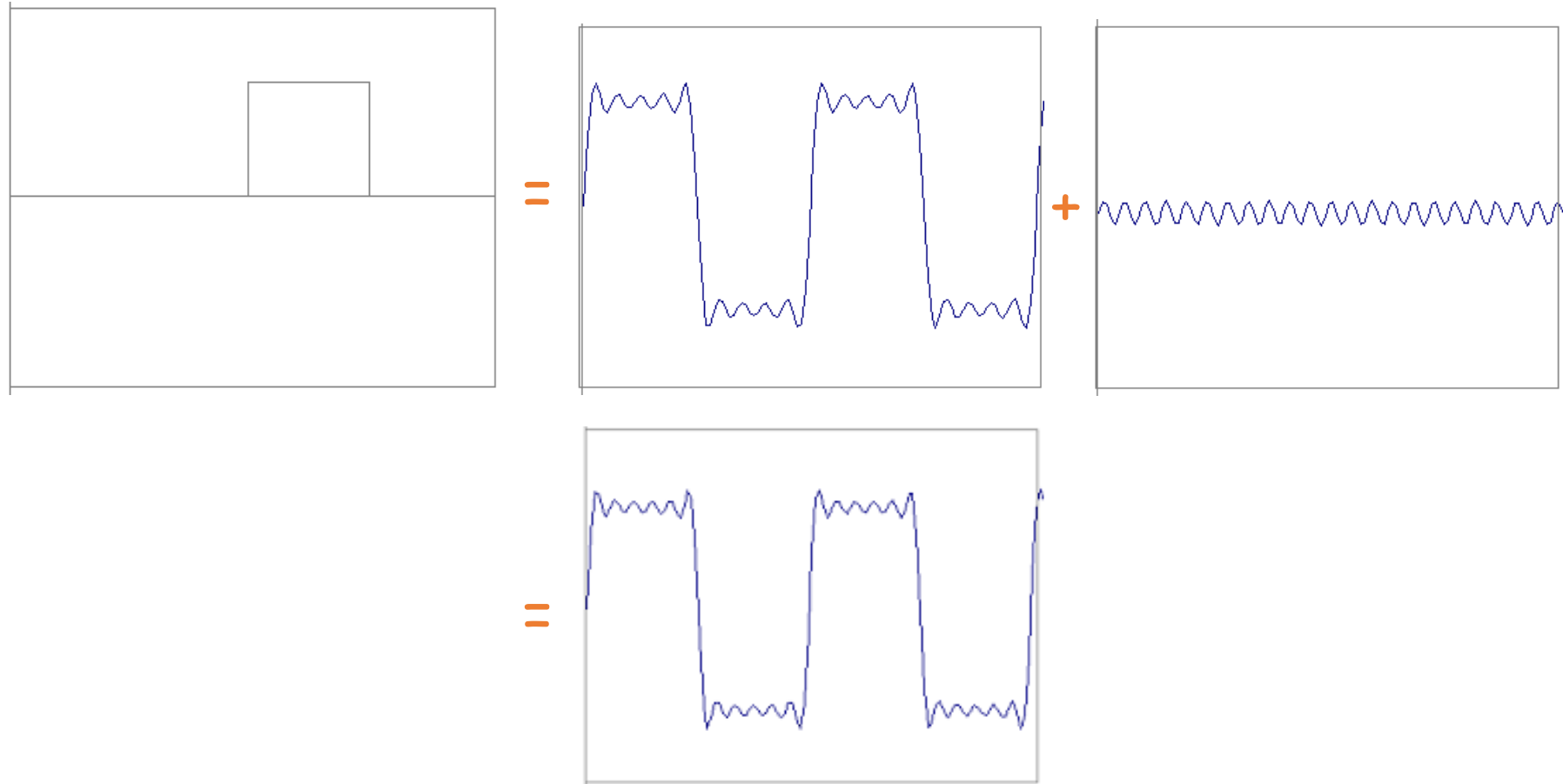
Frequency Spectra



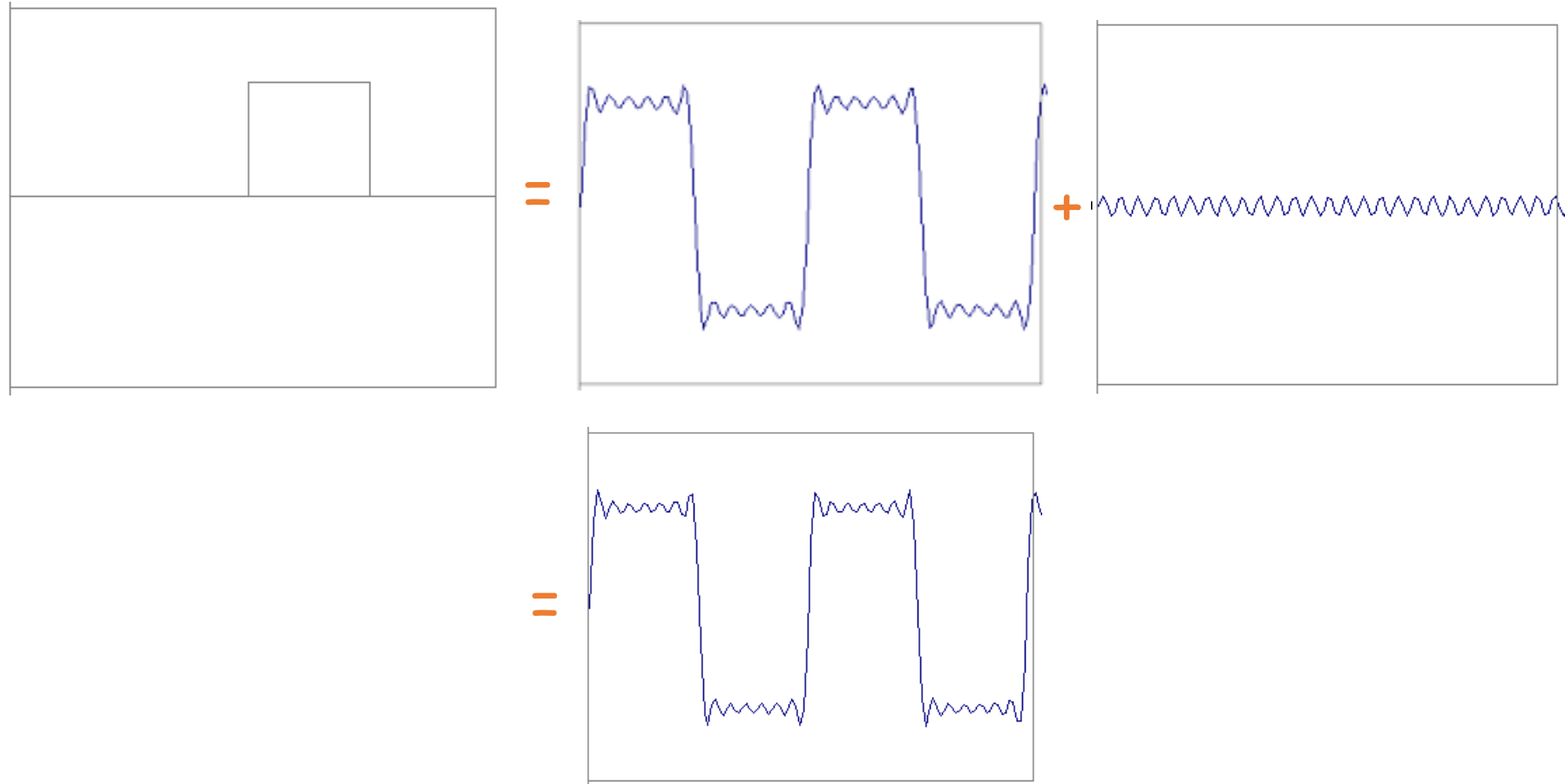
Frequency Spectra



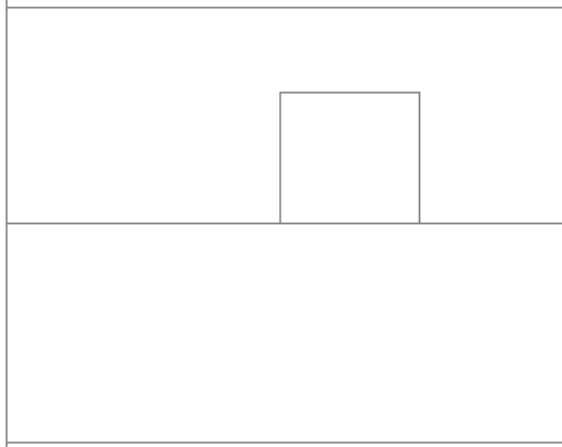
Frequency Spectra



Frequency Spectra

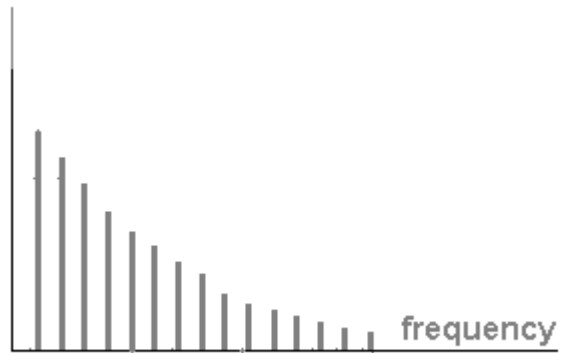


Frequency Spectra

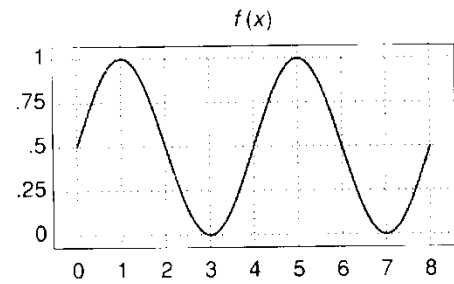


=

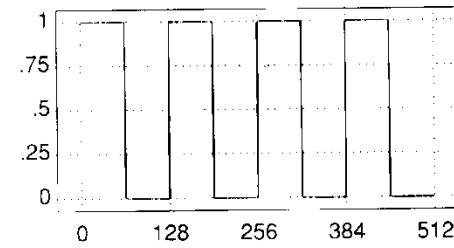
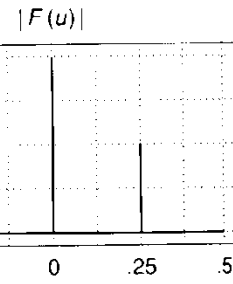
$$A \sum_{k=1}^{\infty} \frac{1}{k} \sin(2\pi kt)$$



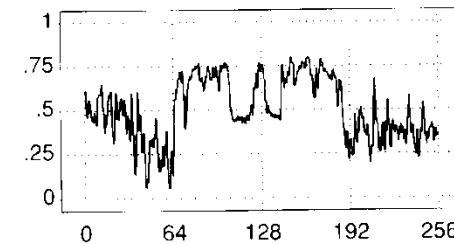
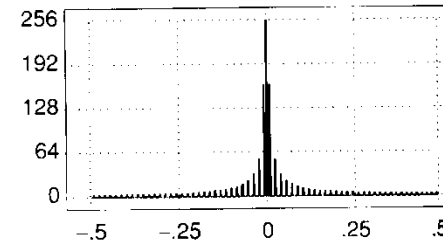
Frequency Spectra



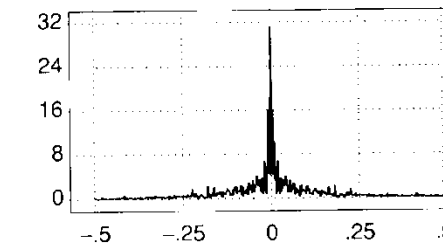
(a)



(b)



(c)



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 ∞ , $F(\omega)$ holds the amplitude A and phase ϕ of the corresponding sine $A \sin(\omega x + \phi)$

- How does F hold both?

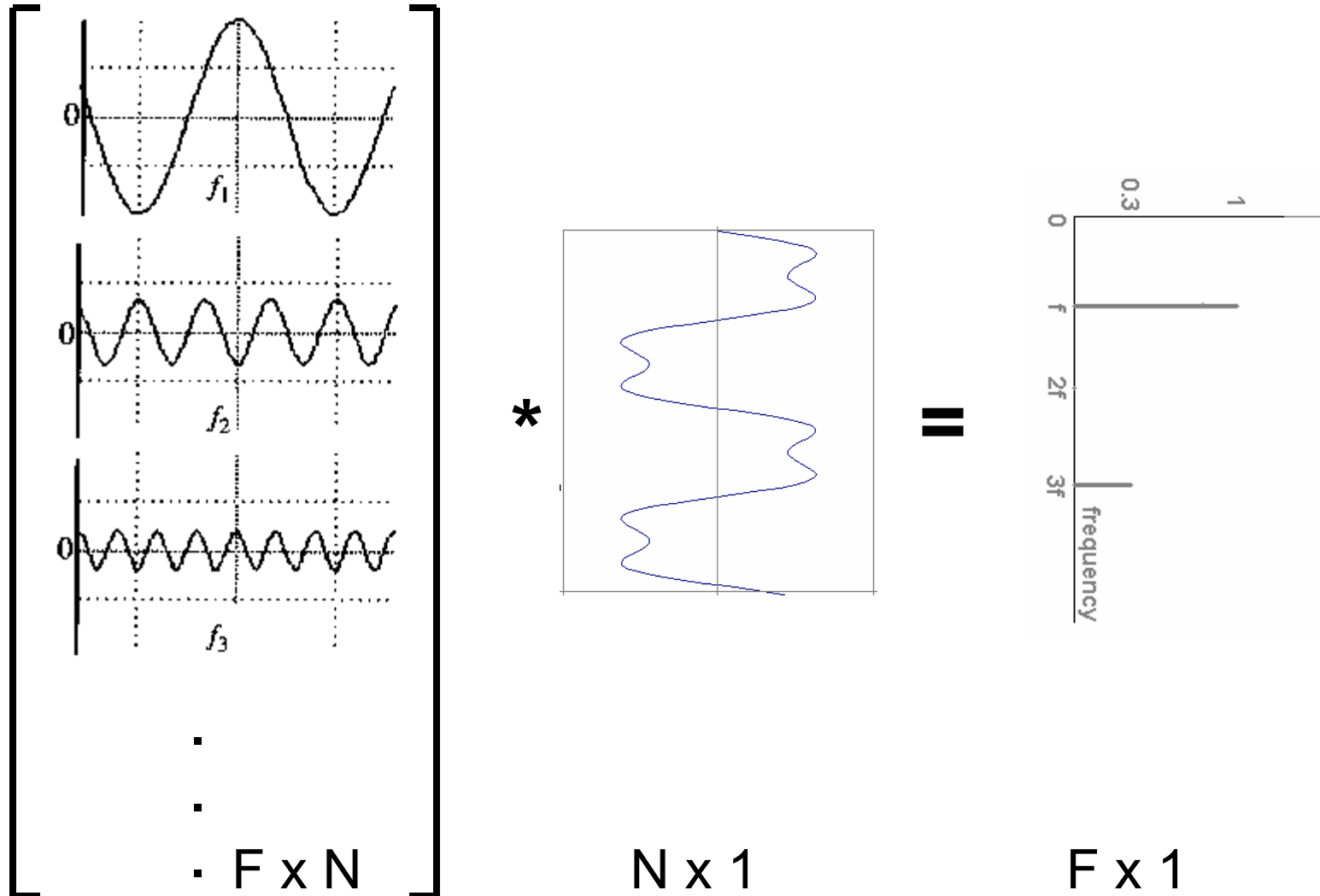
$$F(\omega) = R(\omega) + iI(\omega)$$
$$A = \pm \sqrt{R(\omega)^2 + I(\omega)^2} \quad \phi = \tan^{-1} \frac{I(\omega)}{R(\omega)}$$

We can always go back:



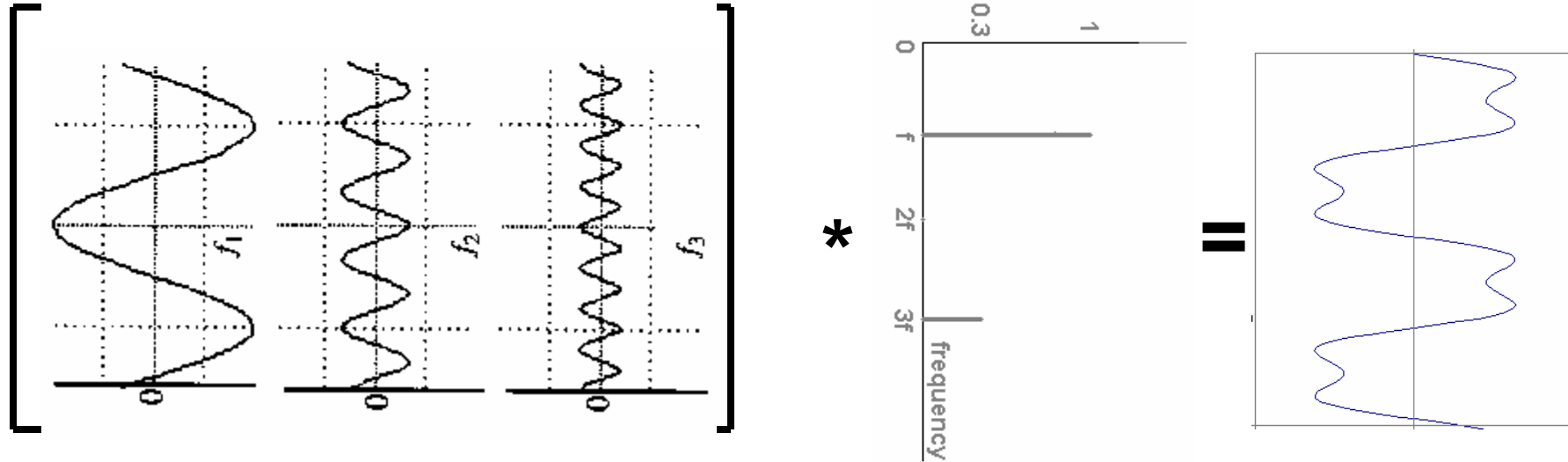
FT: Just a change of basis

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



IFT: Just a change of basis

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



•

•

• $N \times F$

$F \times 1$

$N \times 1$

Finally: Scary Math

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

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

- ...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 \rightarrow

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

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

Discrete Fourier Transform

$$F(\omega) = \int_{-\infty}^{+\infty} f(x)e^{-i\omega x} dx \quad \longrightarrow \quad X[k] = \sum_{n=0}^{N-1} x[n]e^{-j2\pi kn/N}.$$

Discrete Fourier Transform in 2D

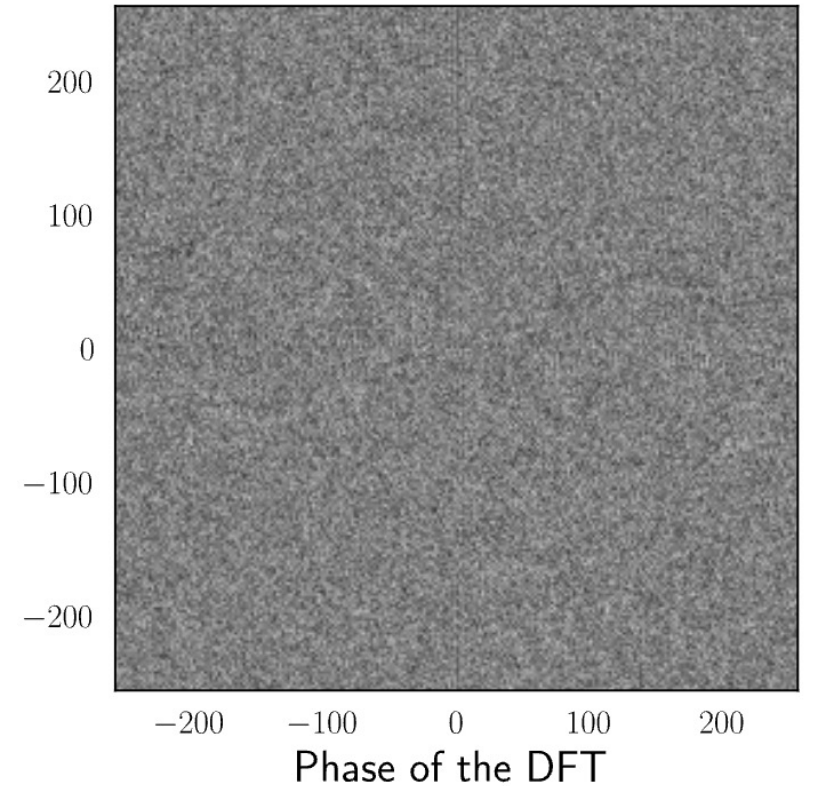
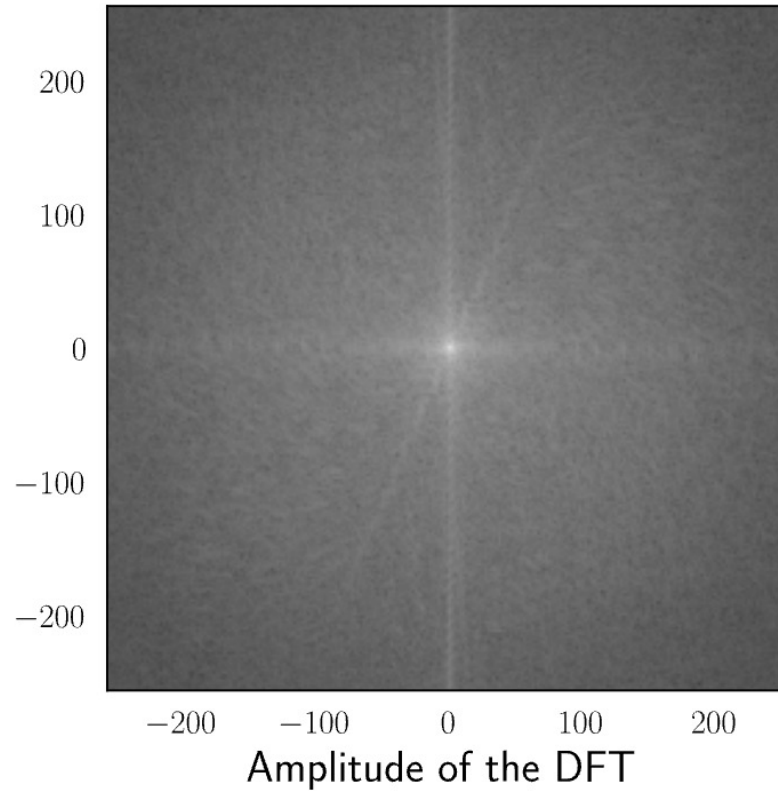
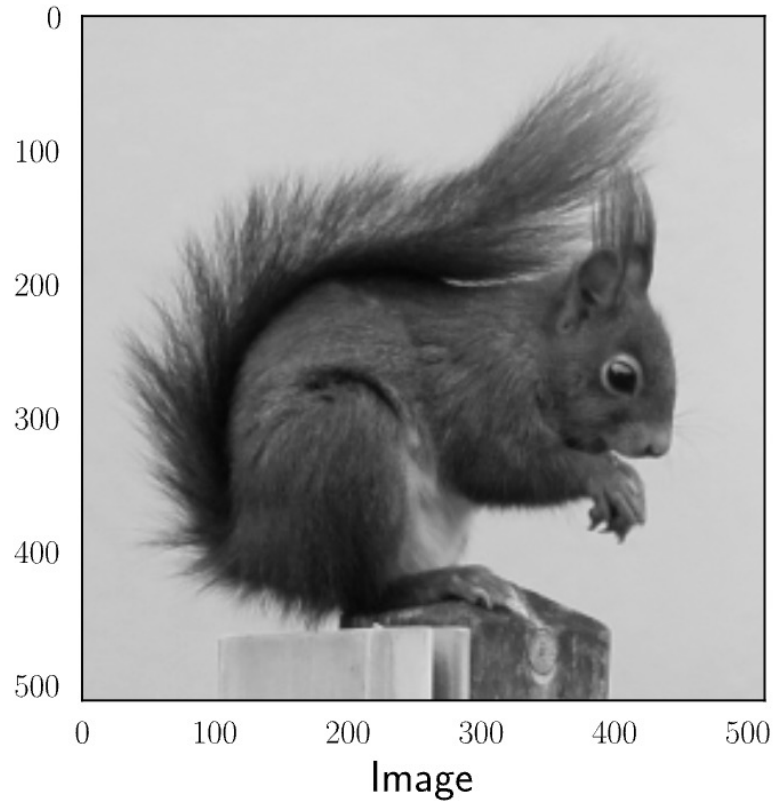
The discrete Fourier transform (DFT) of an image f of size $M \times N$ is an image F of same size defined as:

$$F(u, v) = \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} f(m, n)e^{-j2\pi\left(\frac{um}{M} + \frac{vn}{N}\right)}$$

Today's Lecture

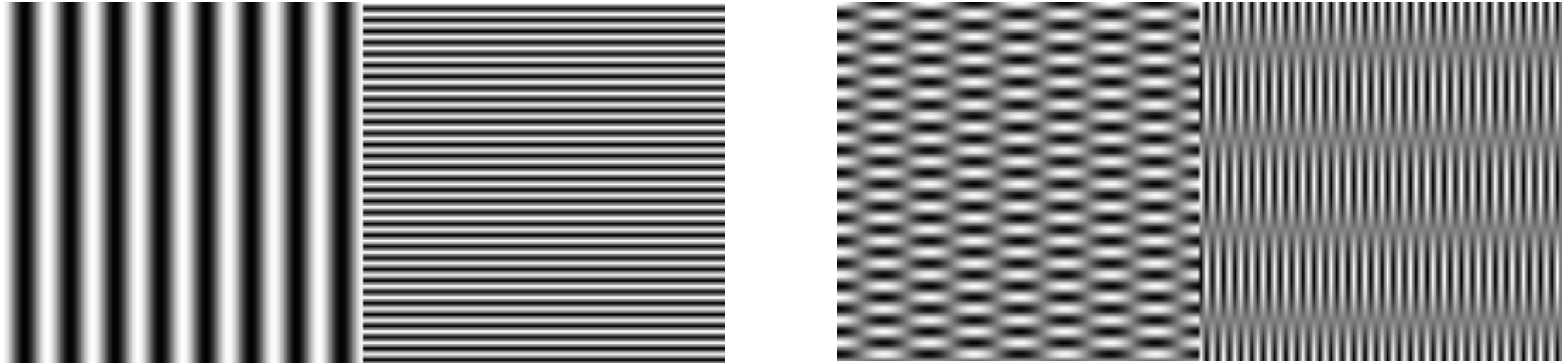
- Image Derivatives
- Edge Detection
- Fourier Analysis (in 1D)
- **Fourier Analysis (in 2D)**

Discrete Fourier Transform (DFT) in 2D



Amplitude Spectrum of DFT

Images

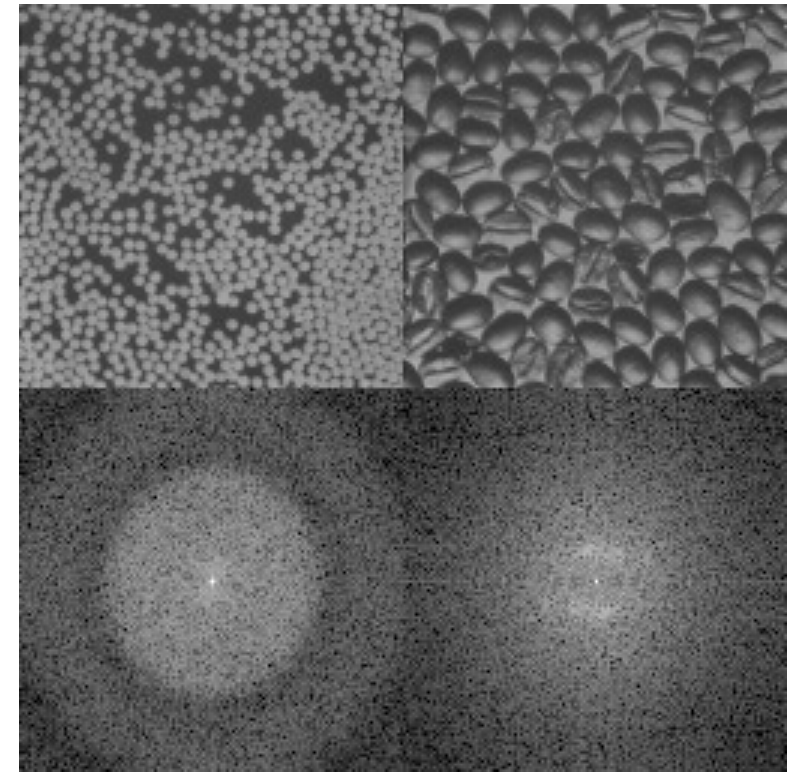
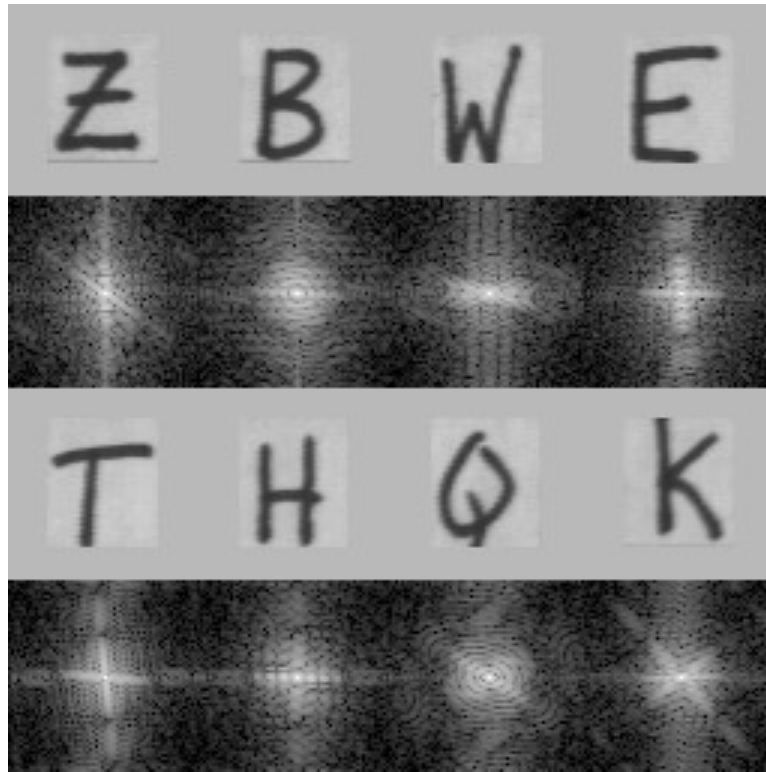
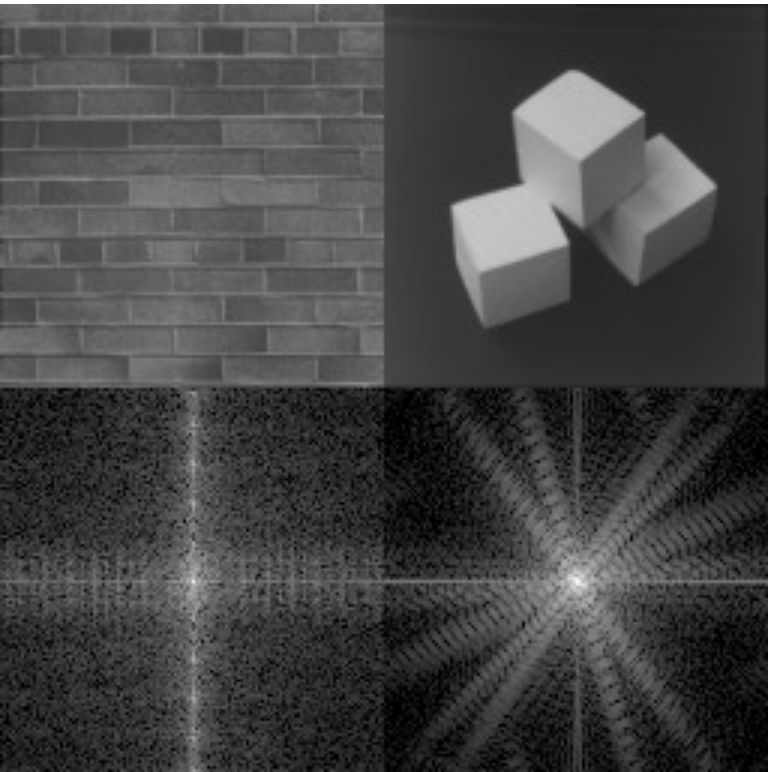


Amplitude spectrum



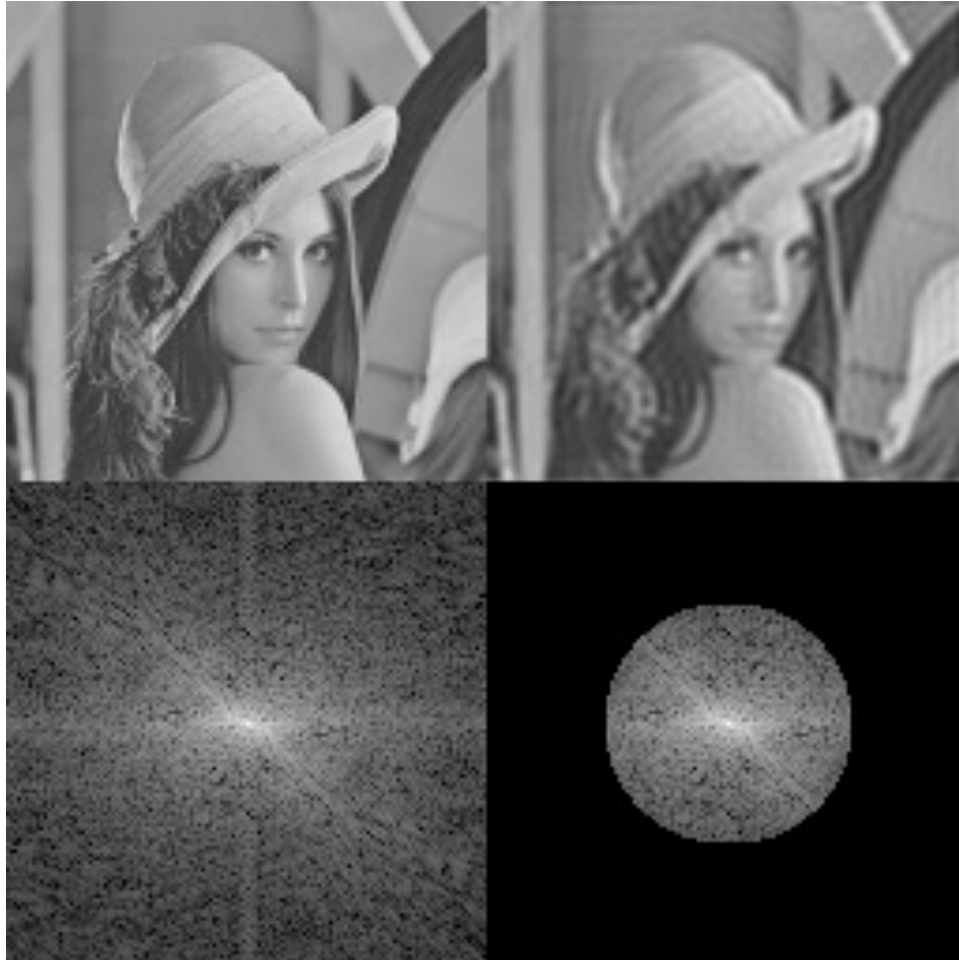
horizontal change (vertical stripes) in image -> frequency response in x-axis

Amplitude Spectrum of DFT

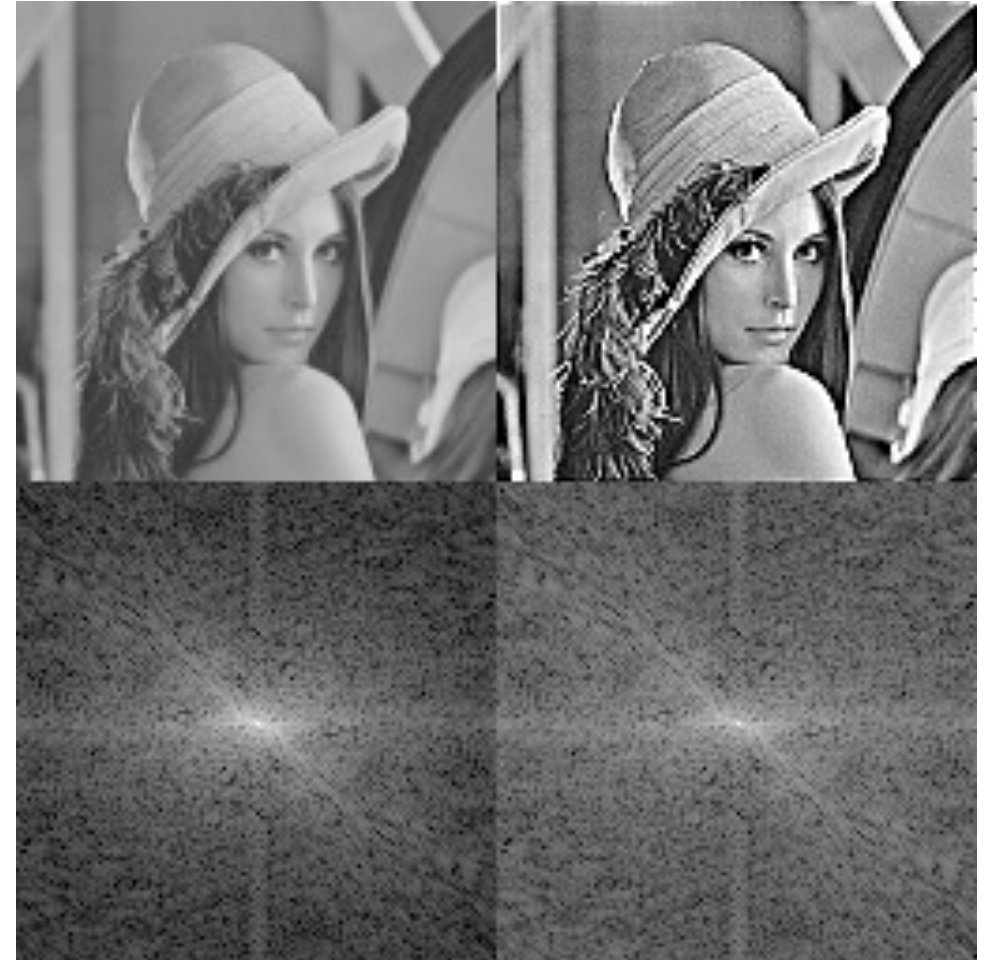


Amplitude spectrum indicates distribution of edge direction in the images

Filtering & Frequency Domain

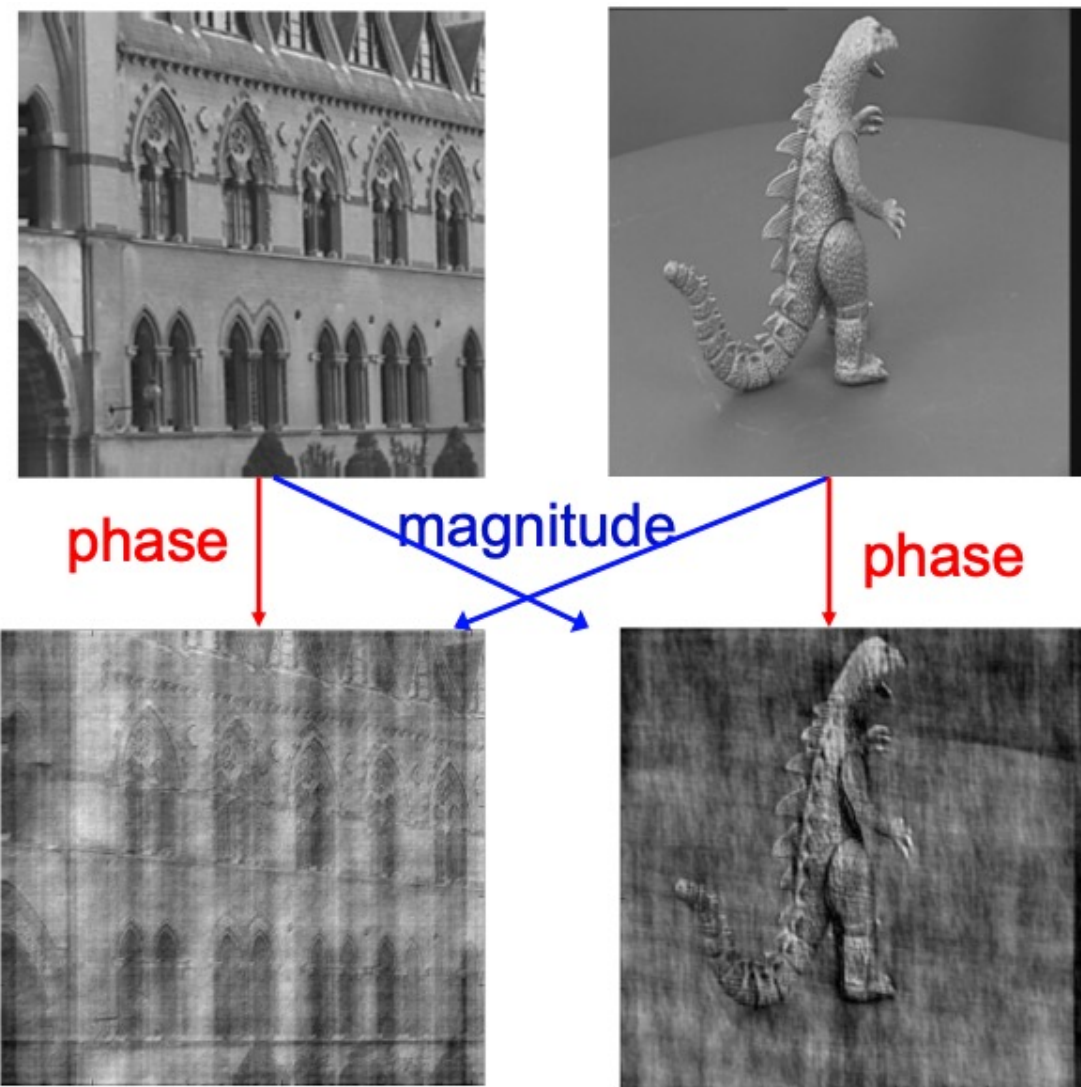


Low pass Filtering with Gaussian Filter



High pass Filtering with Laplacian Filter

The importance of Phase



Amplitude doesn't carry the image information. Infact the phase is one that contains image information and is very important for reconstruction.

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

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

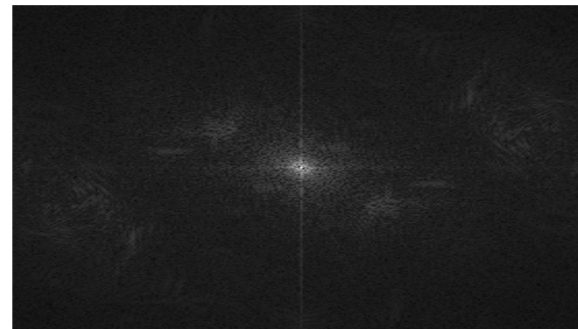
- **Convolution** in spatial domain is equivalent to **multiplication** in frequency domain!

2D convolution theorem example

$f(x,y)$



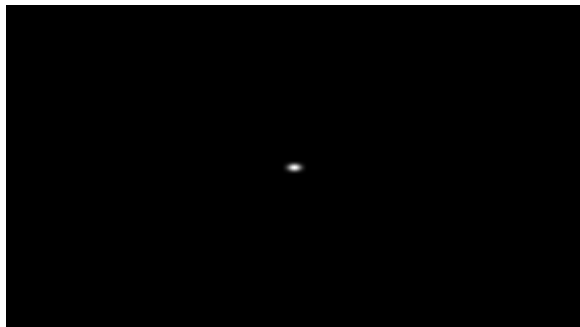
*



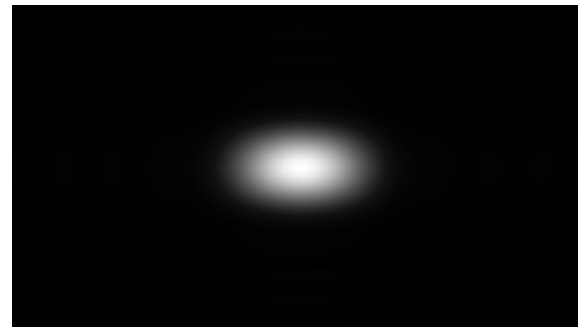
$|F(s_x, s_y)|$

×

$h(x,y)$



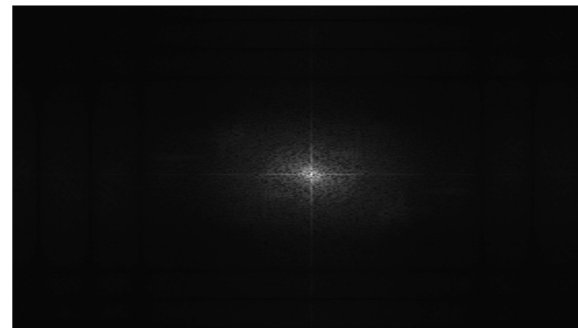
⇓



$|H(s_x, s_y)|$

⇓

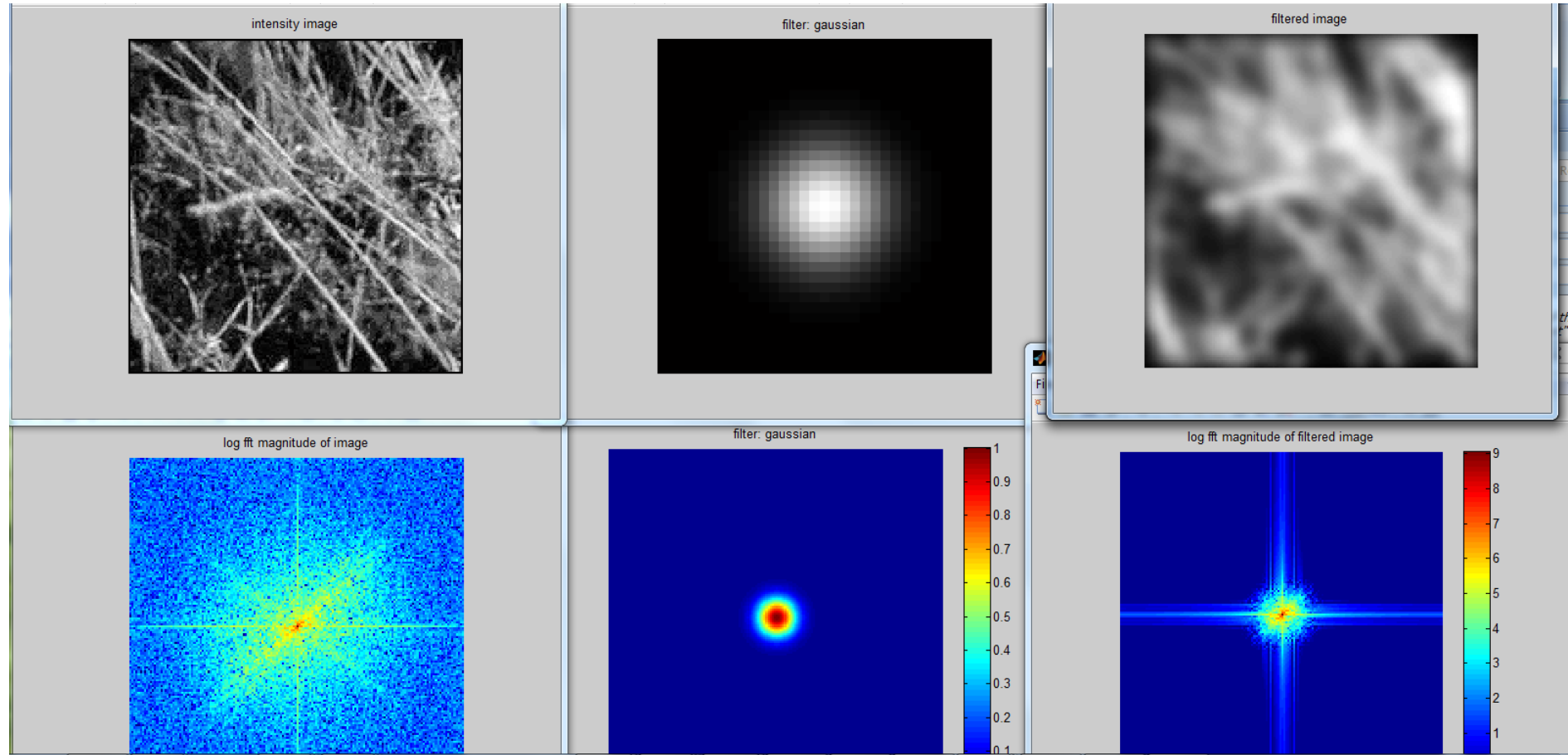
$g(x,y)$



$|G(s_x, s_y)|$

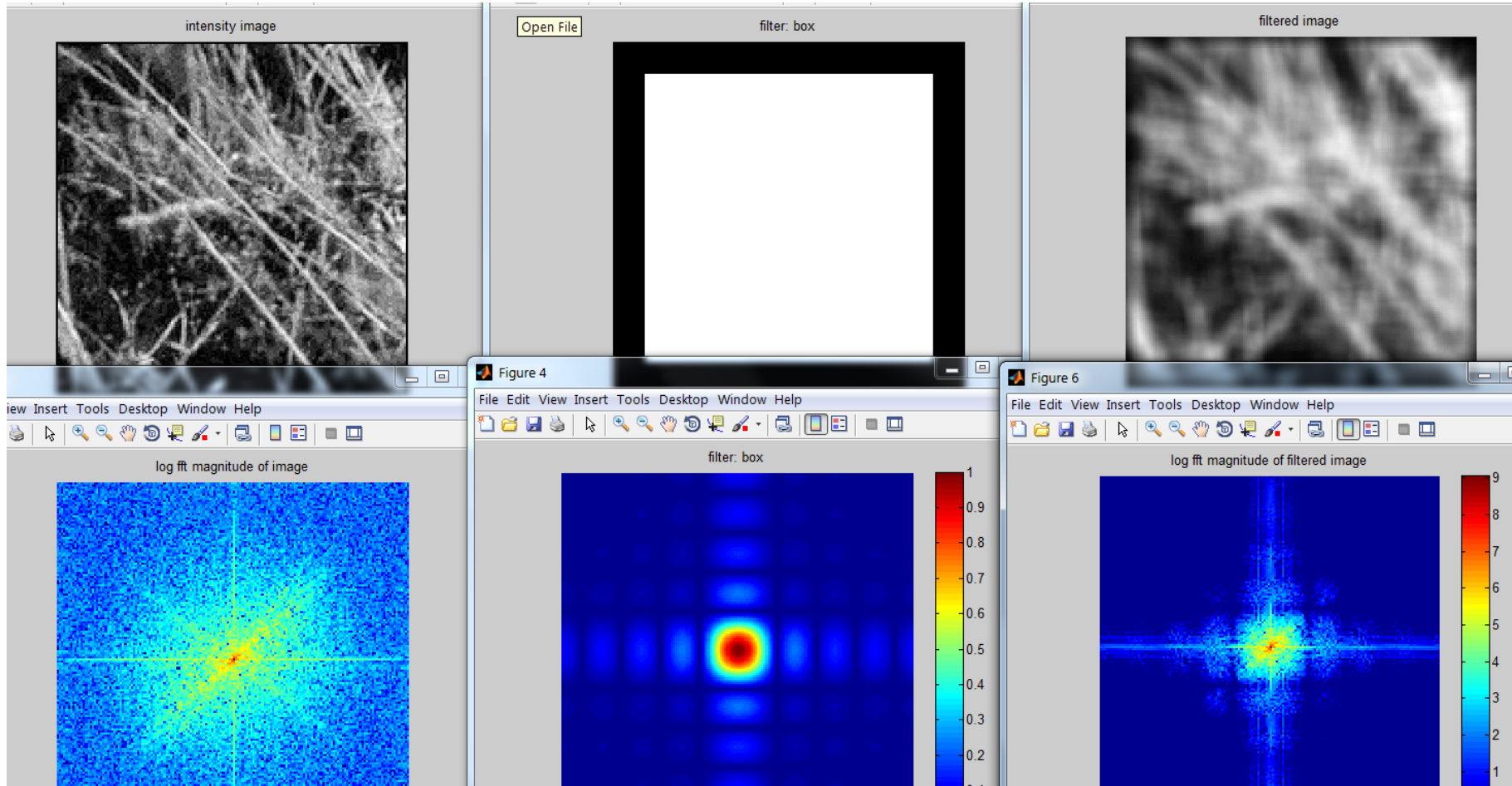
Why does the Gaussian give a nice smooth image, but the square filter give edgy artifacts?

Gaussian



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

Box Filter



Fourier Transform pairs

Spatial domain

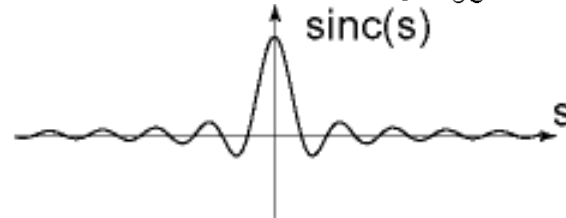
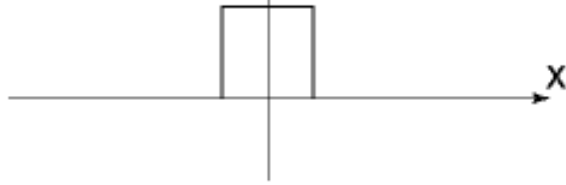
Frequency domain

$f(x)$

$$F(s) = \int_{-\infty}^{\infty} f(x) e^{-i2\pi s x} dx$$

↑ box(x)

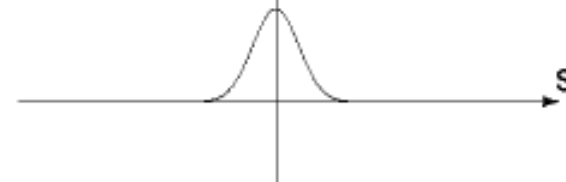
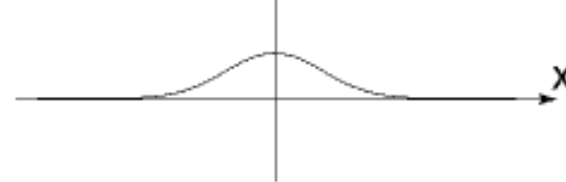
↑ sinc(s)



Causes aliasing!

↑ gauss(x; σ)

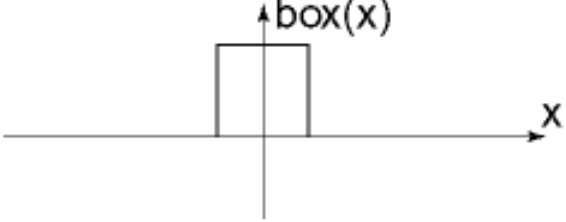
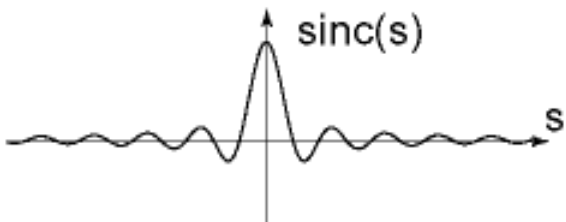
↑ gauss(s; $1/\sigma$)



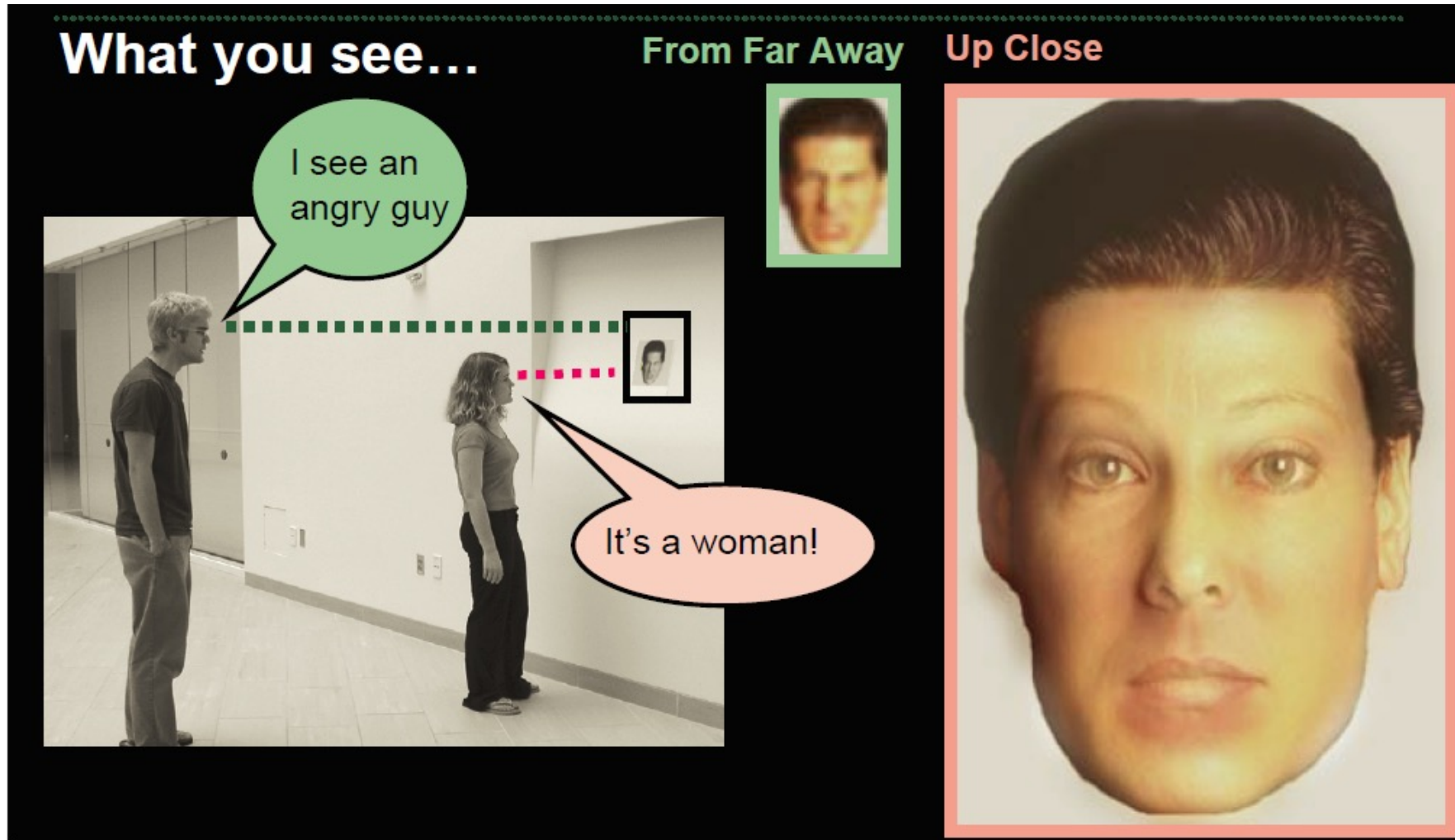
What is the best low pass filter?

↑ sinc(s)

↑ box(x)

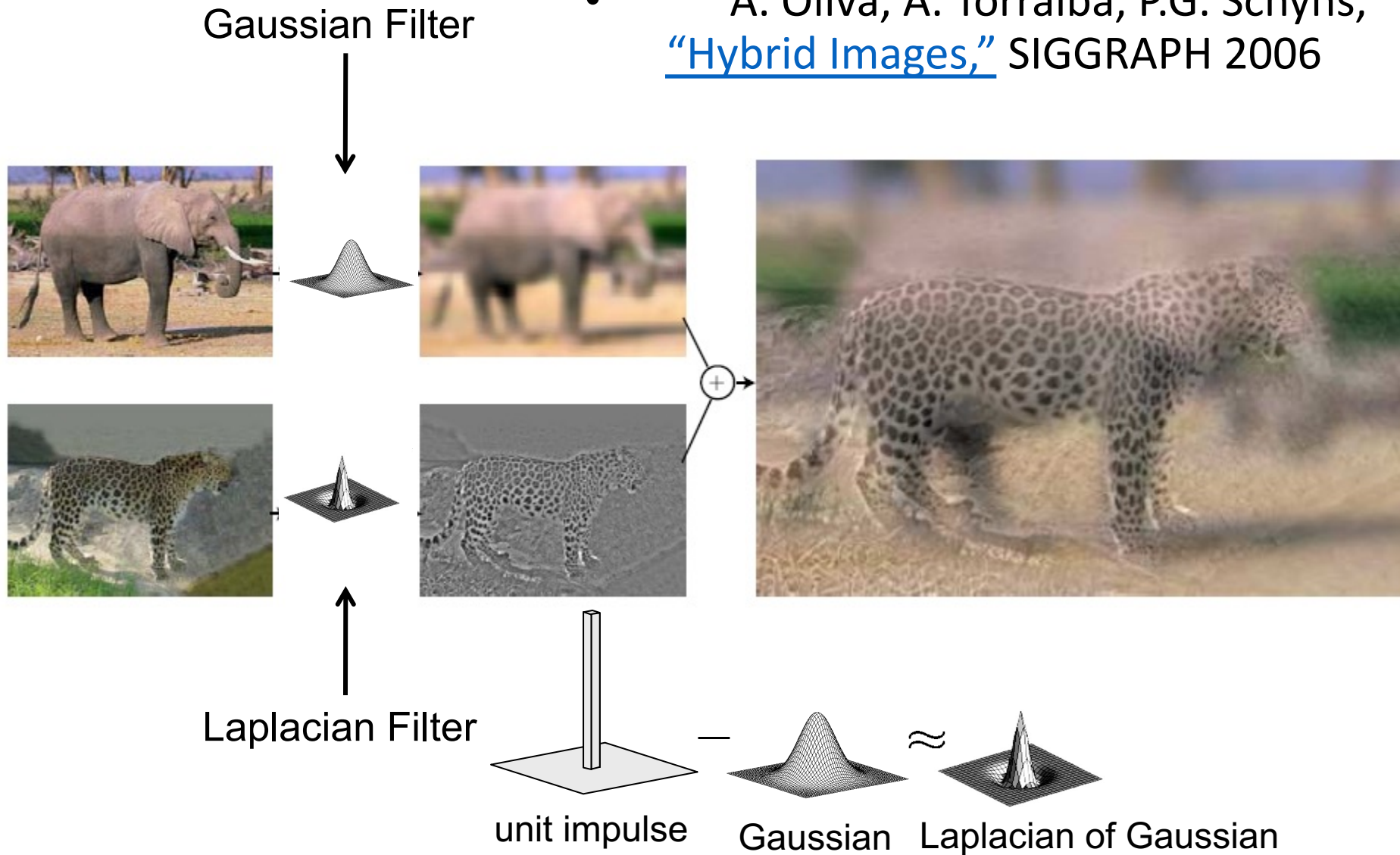


Application: Hybrid Images (in HW)



Application: Hybrid Images

- A. Oliva, A. Torralba, P.G. Schyns, [“Hybrid Images,”](#) SIGGRAPH 2006




Fourier and frequency is 18th century stuff!
I wanna learn Deep Learning, train neural nets,
beat the benchmark by 5% and publish in CVPR!

$$\gamma(\mathbf{v}) = [\cos(2\pi\mathbf{B}\mathbf{v}), \sin(2\pi\mathbf{B}\mathbf{v})]^T$$


 **Jon Barron**
@jon_barron

Three years ago today, the project that eventually became NeRF started working (positional encoding was the missing piece that got us from "hmm" to "wow"). Here's a snippet of that email thread between Matt Tancik, @_pratul_, @BenMildenhall, and me. Happy birthday NeRF!

 **Matthew Tancik** 1/17/2020
to me, Pratul, BEN ▾

After you left we did some experiments where we augmented our inputs with the multiscale sin/cos values that we discussed, ie. input = [sin(coordinate * 2ⁱ) for i in range(x)]. the 2d toy problem we found a significant boost in performance (see attached images). We are currently doing experiments to see if we also see improvements in the 3D case, or if we run into generalization issues.



 **Jon Barron** 1/17/2020
to Matthew, Pratul, BEN ▾

holy shit!

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