Computer Vision

Doctoral Program in Computer Science (MAPi)

Hélder Filipe Pinto de Oliveira

Faculdade de Ciências da Universidade do Porto
Departamento de Ciência de Computadores

DOCTORAL PROGRAMME
IN COMPUTER SCIENCE

Outline

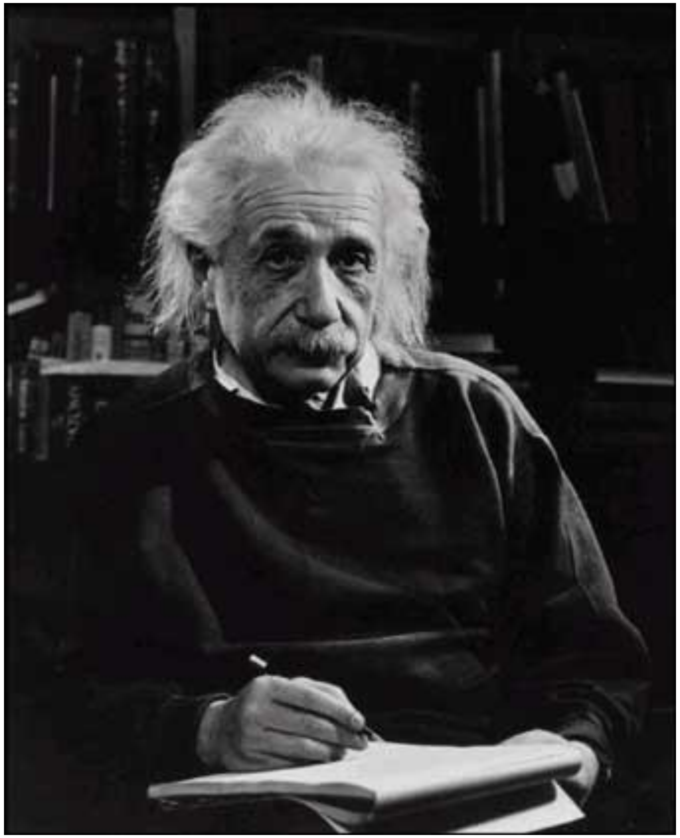
- Single Pixel Manipulation
- Frequency Space
- Digital Filters

Acknowledgements: Most of this course is based on the excellent courses offered by Prof. Shree Nayar at Columbia University, USA and by Prof. Srinivasa Narasimhan at CMU, USA. Please acknowledge the original source when reusing these slides for academic purposes.

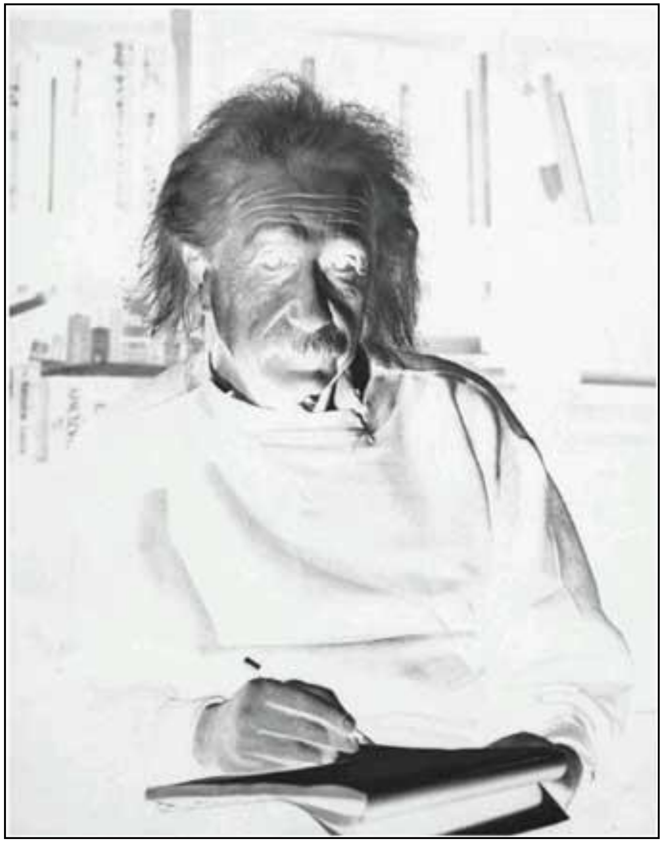
Outline

- Single Pixel Manipulation
  - Dynamic Range Manipulation
  - Neighborhoods and Connectivity
  - Image Arithmetic
  - Example: Background Subtraction
- Frequency Space
- Digital Filters

Manipulation

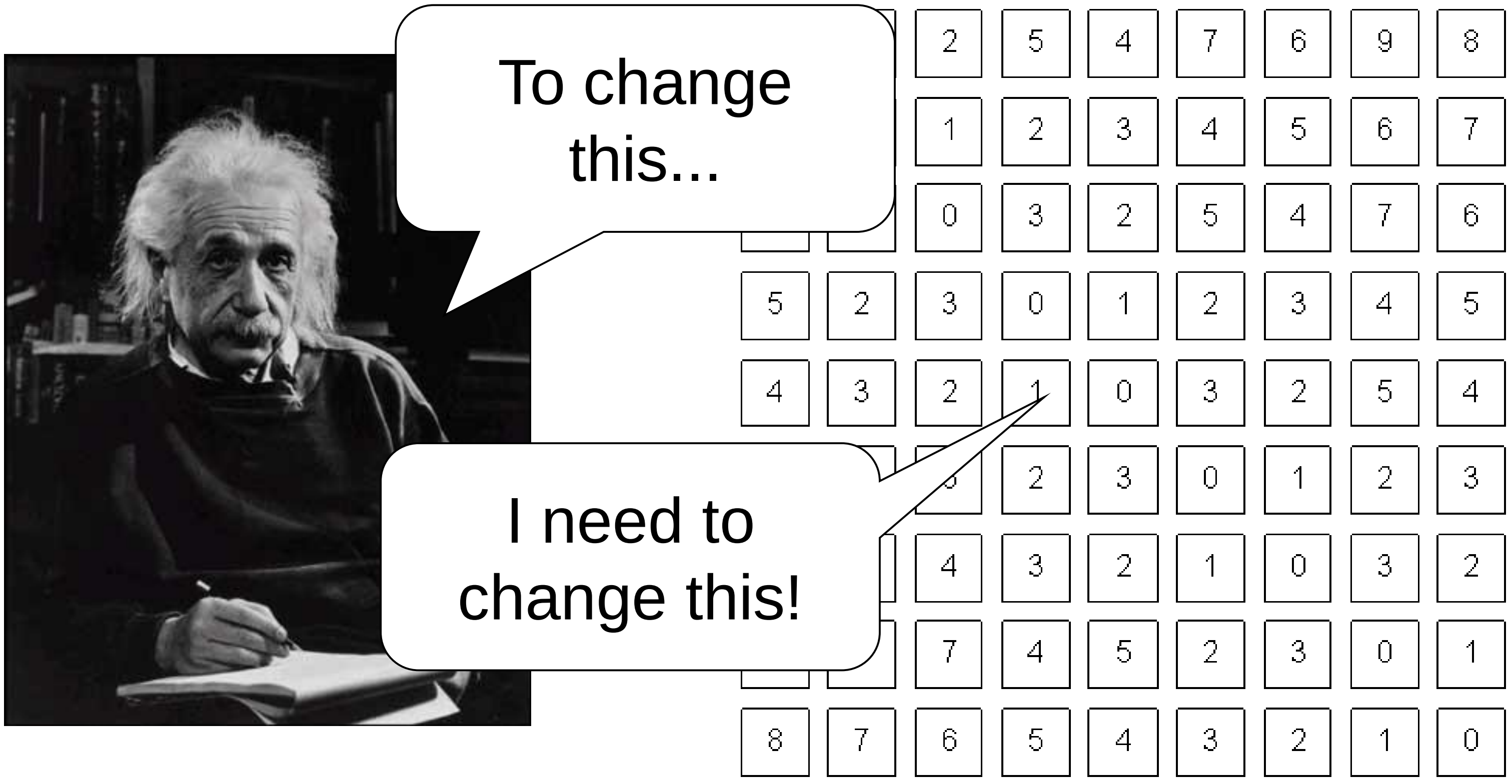


What I see



What I want to see

Digital Images



What we see

What a computer sees

Pixel Manipulation

- Let’s start simple
- I want to change a single Pixel.

$$f(X,Y) = MyNewValue$$

- Or, I can apply a transformation T to all pixels individually.

$$g(x,y) = T\left[f(x,y)\right]$$

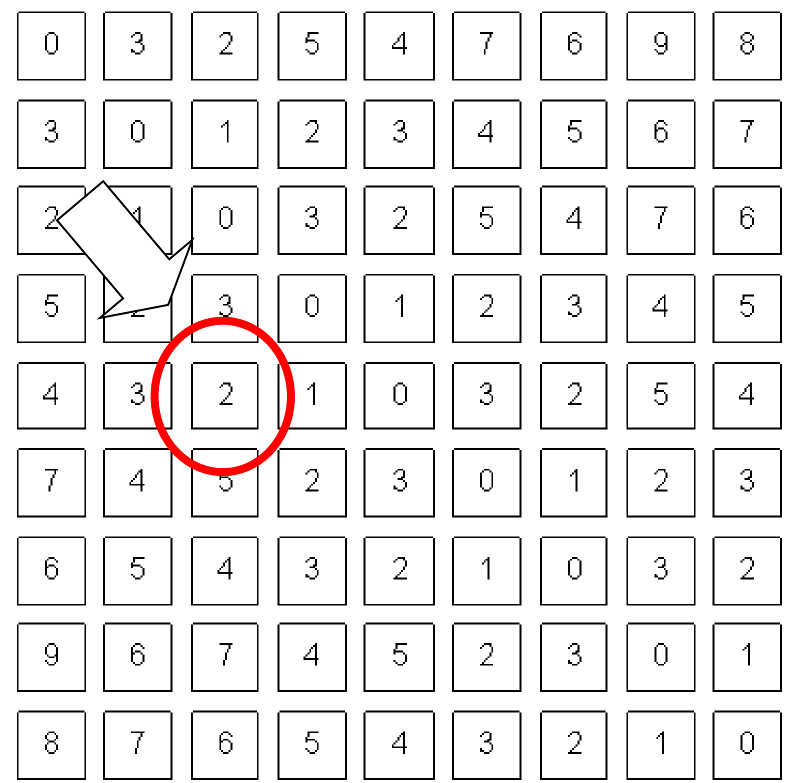


Image Domain (Spatial)

- I am directly changing values of the image matrix.

$$g = T(f)$$

- Image Domain
- So, what is the other possible ‘domain’?

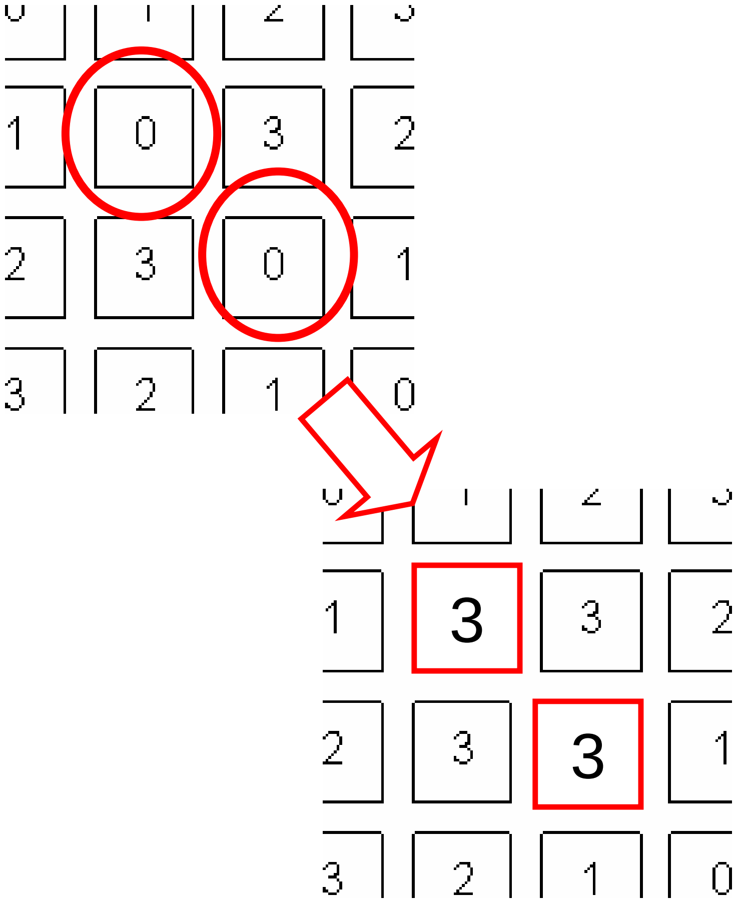
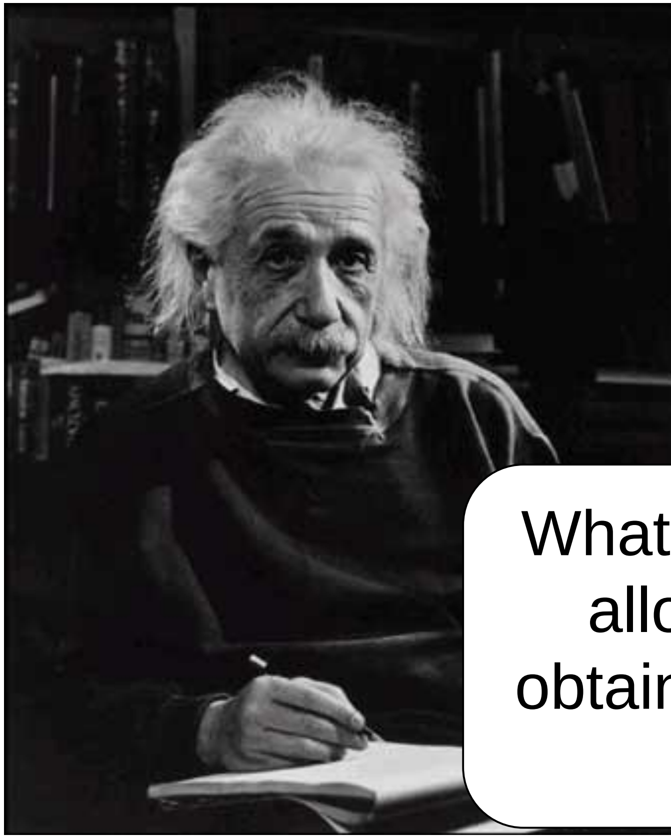


Image Negative



What I see

What I want to see

What operation T allows me to obtain this result?

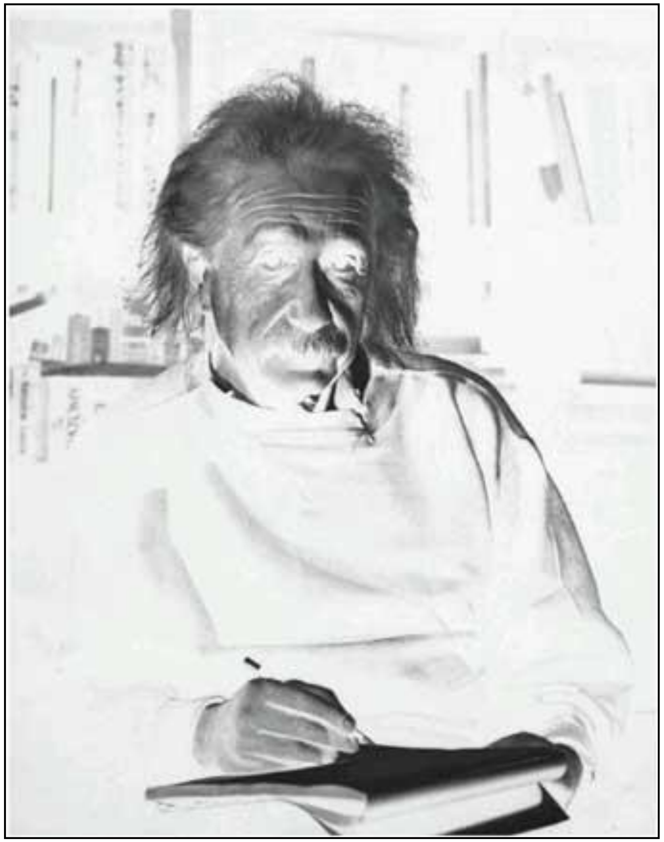
$g=T(f)$

Image Negative

- Consider the maximum value allowed by quantization (*max).*
- For 8 bits: 255
- Then:

$$g(x, y) = \max - f(x, y)$$

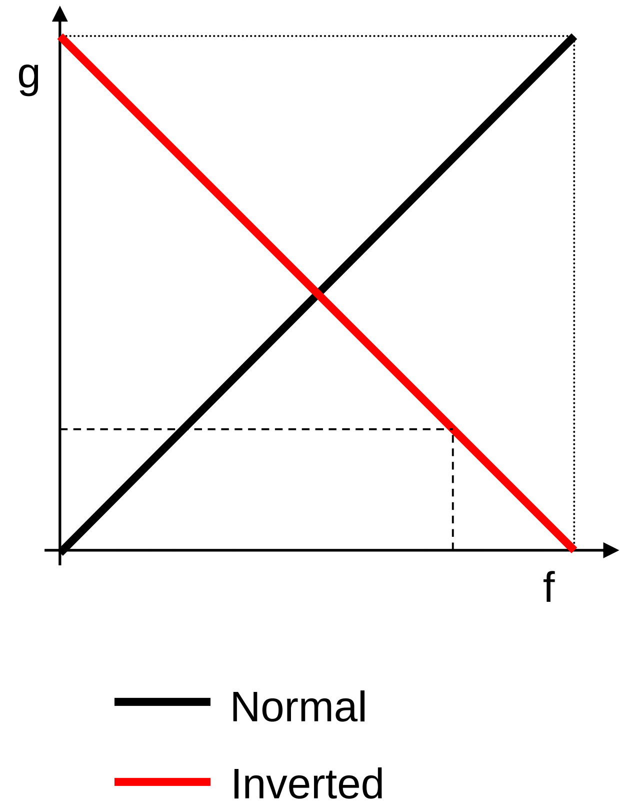
$$g(x, y) = 255 - f(x, y)$$



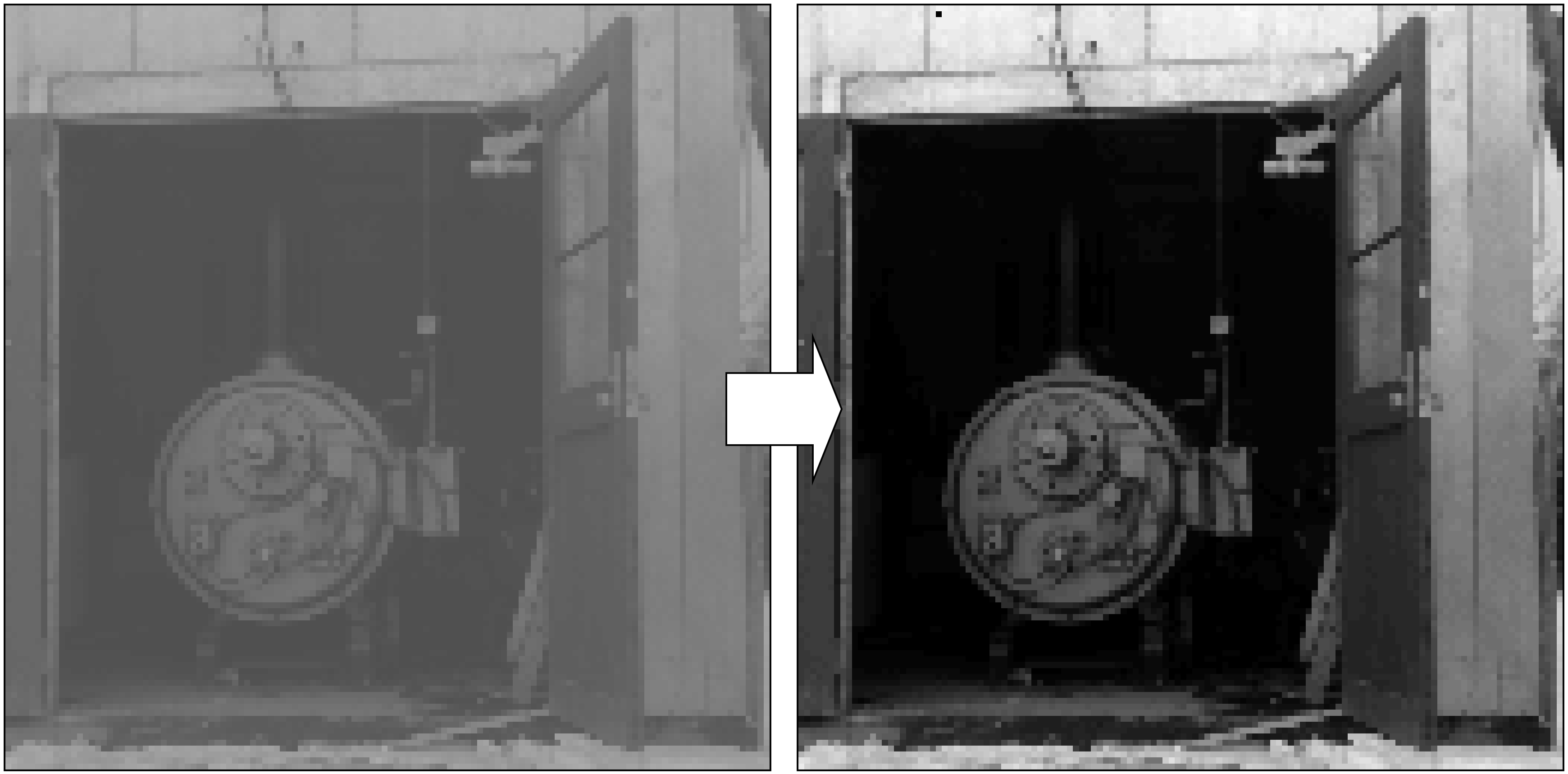
What I <u>want</u> to see

Dynamic Range Manipulation

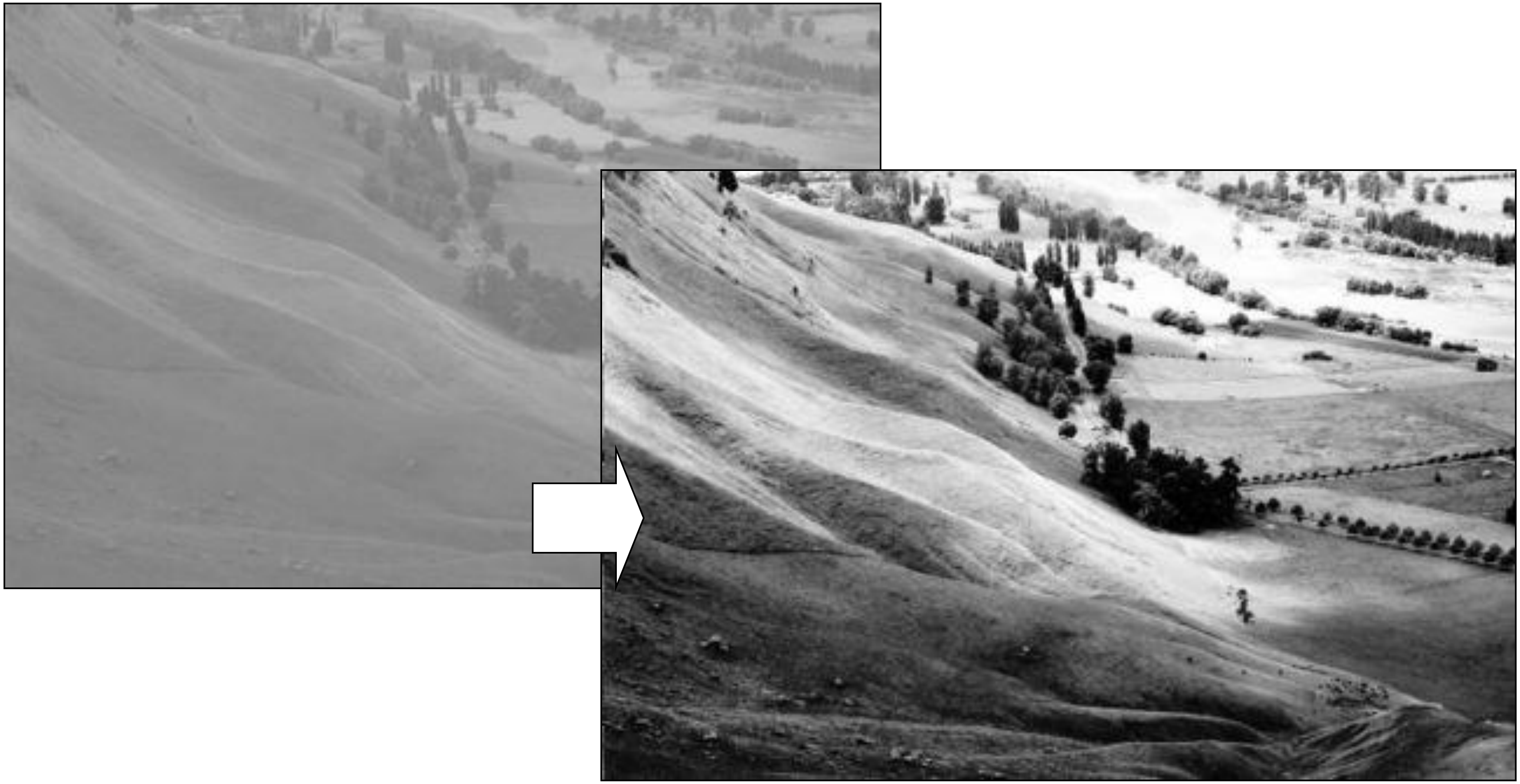
- What am I really doing?
  - Changing the response of my image to the received brightness.
- Dynamic Range Manipulation
  - Represented by a 2D Plot.



Why DRM?

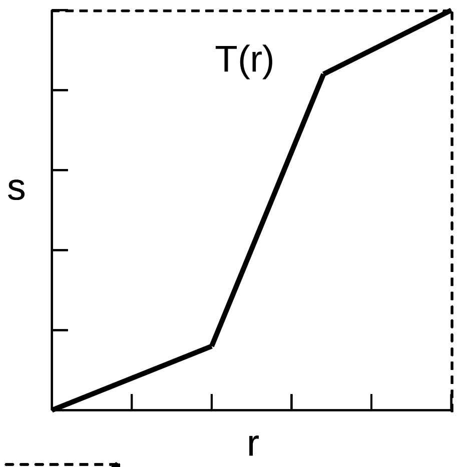


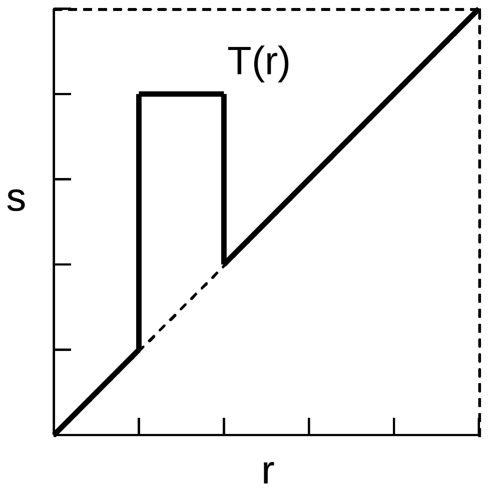
Why DRM?



Other DRM functions

- By manipulating our function we can:
  - Enhance generic image visibility.
  - Enhance specific visual features.
  - Use quantization space a lot better.

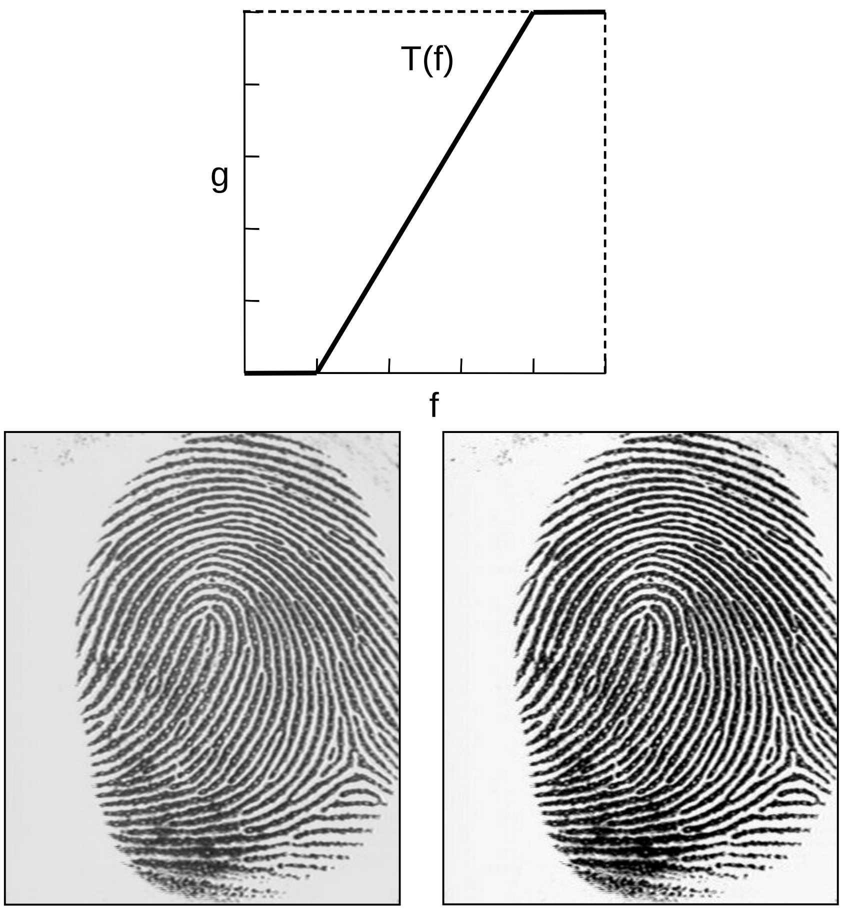




Contrast Stretching

- ‘Stretches’ the dynamic range of an image.
- Corrects some image capture problems:
  - Poor illumination, aperture, poor sensor performance, etc.

$$g = 255 \frac{f - \min}{\max - \min}$$



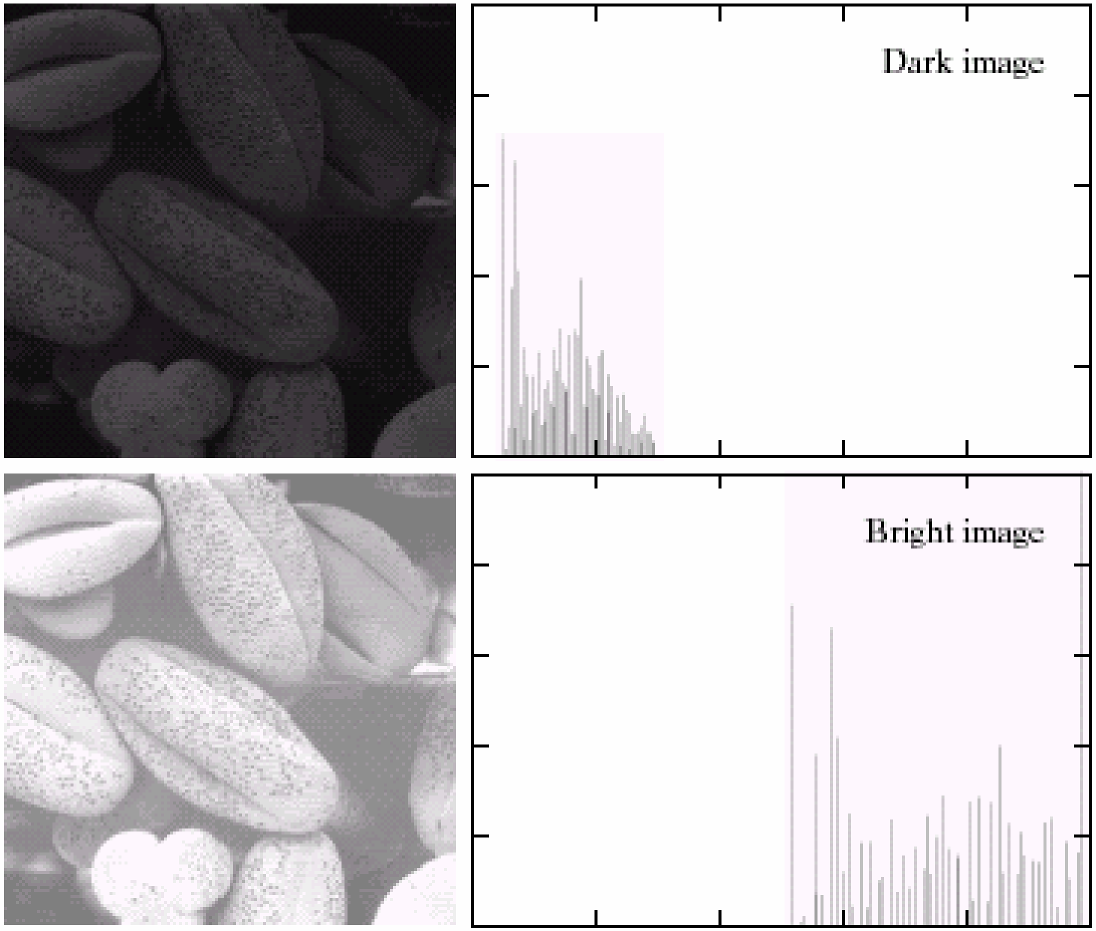
Histogram Processing

- Histograms give us an idea of how we are using our dynamic range

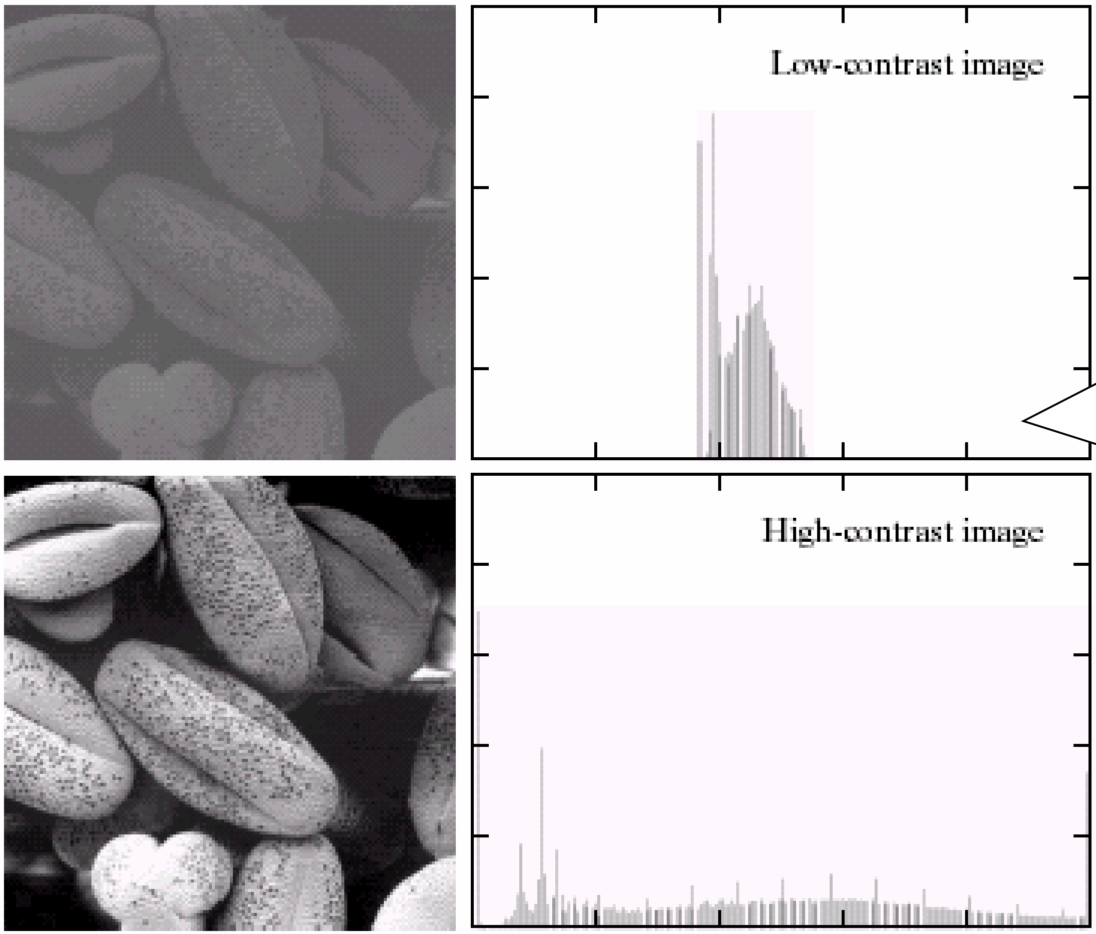
What if I want a different histogram for my image?

Types of Image Histograms

- Images can be classified into types according to their histogram
  - Dark
  - Bright
  - Low-contrast
  - High-contrast



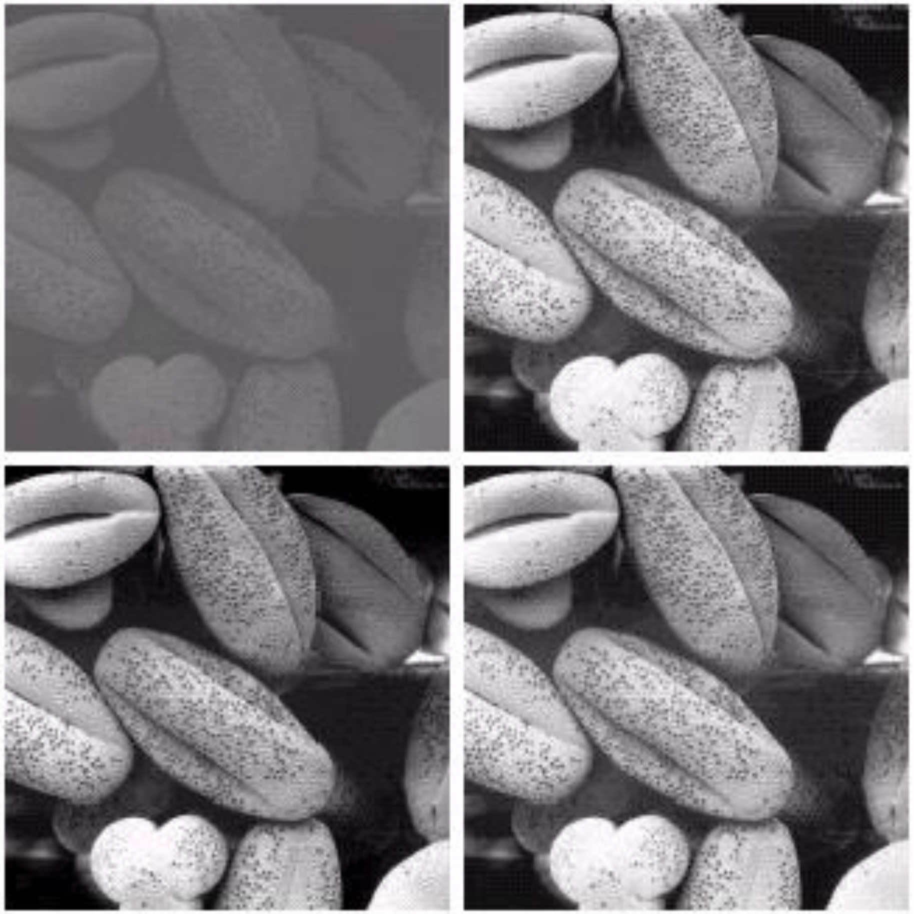
Types of Image Histograms



I can manipulate this using single Pixel operations!

Histogram Equalization

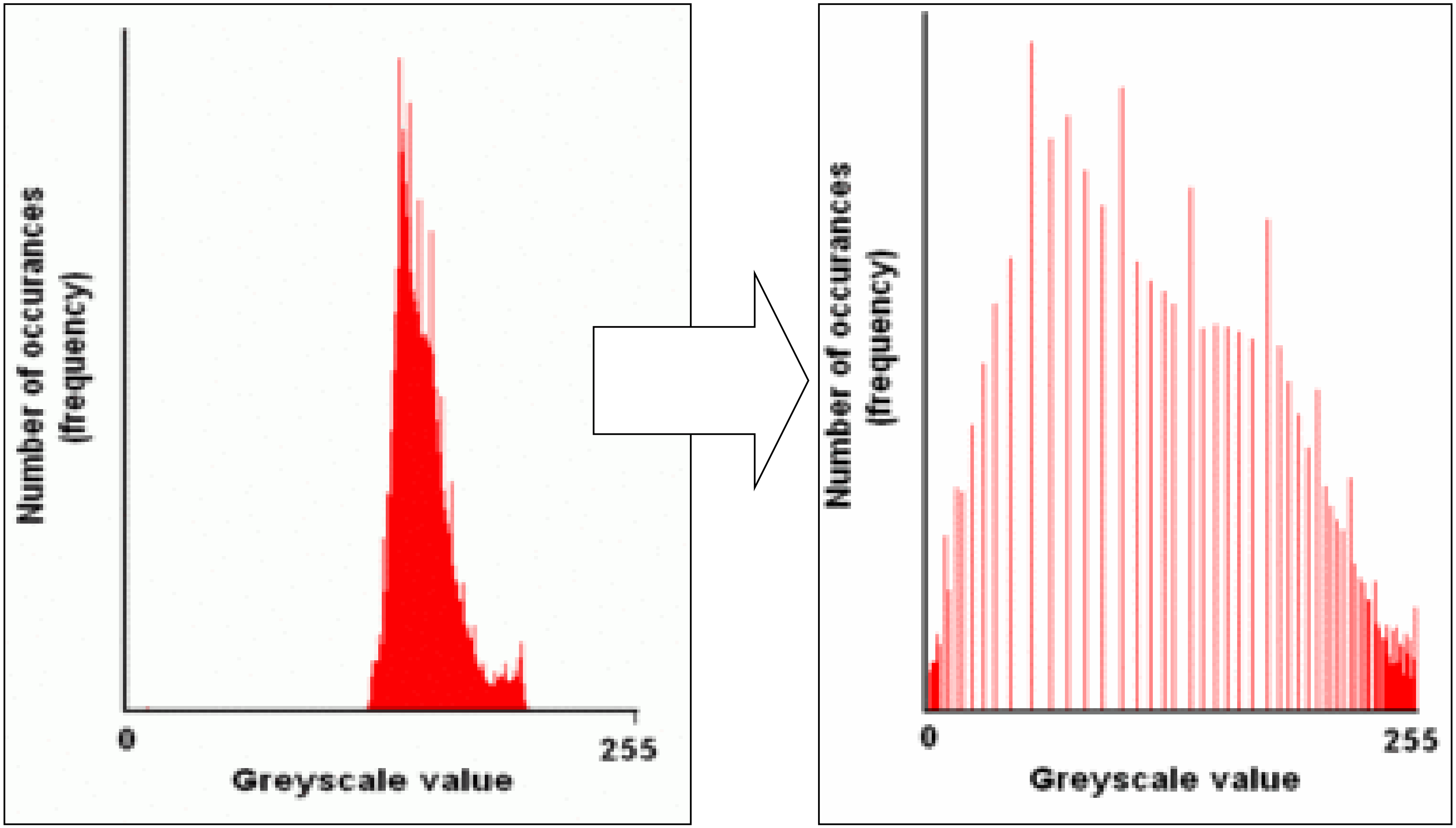
$$s_k = T(r_k) = \sum_{j=0}^{k} \frac{n_j}{n} = \sum_{j=0}^{k} p_r(r_j)$$



- Objective:
  - Obtain a ‘flat’ histogram.
  - Enhance visual contrast.
- Digital histogram
  - Result is a ‘flat-ish’ histogram.
  - Why?

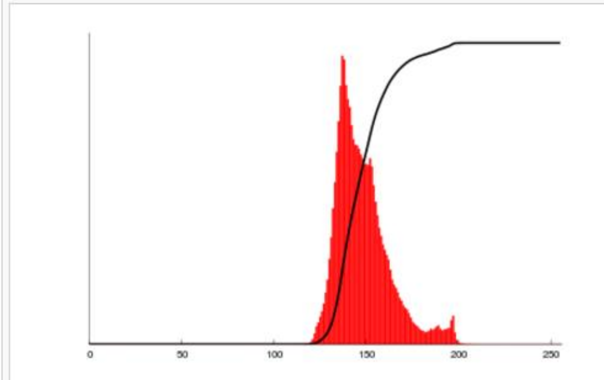
https://en.wikipedia.org/wiki/Histogram_equalization

Histogram Equalization



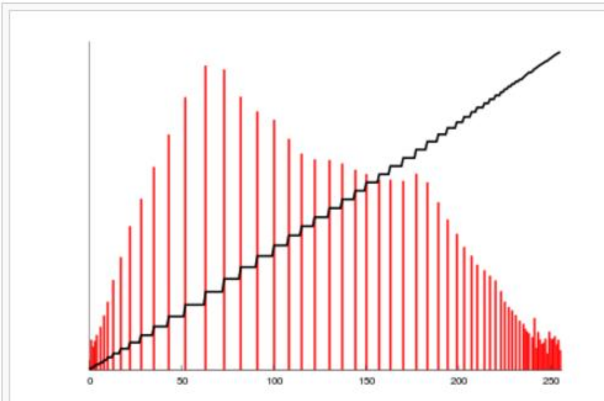
Histogram Equalization

An unequalized image



Corresponding histogram (red) and cumulative histogram (black)

The same image after histogram equalization



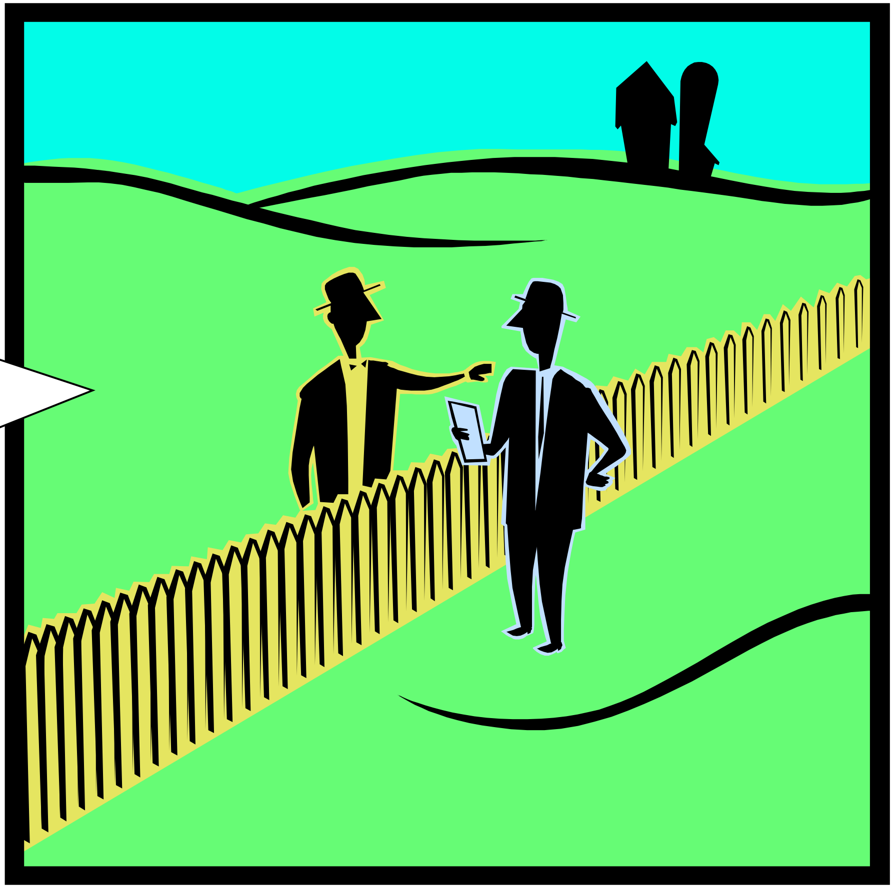
Corresponding histogram (red) and cumulative histogram (black)

Outline

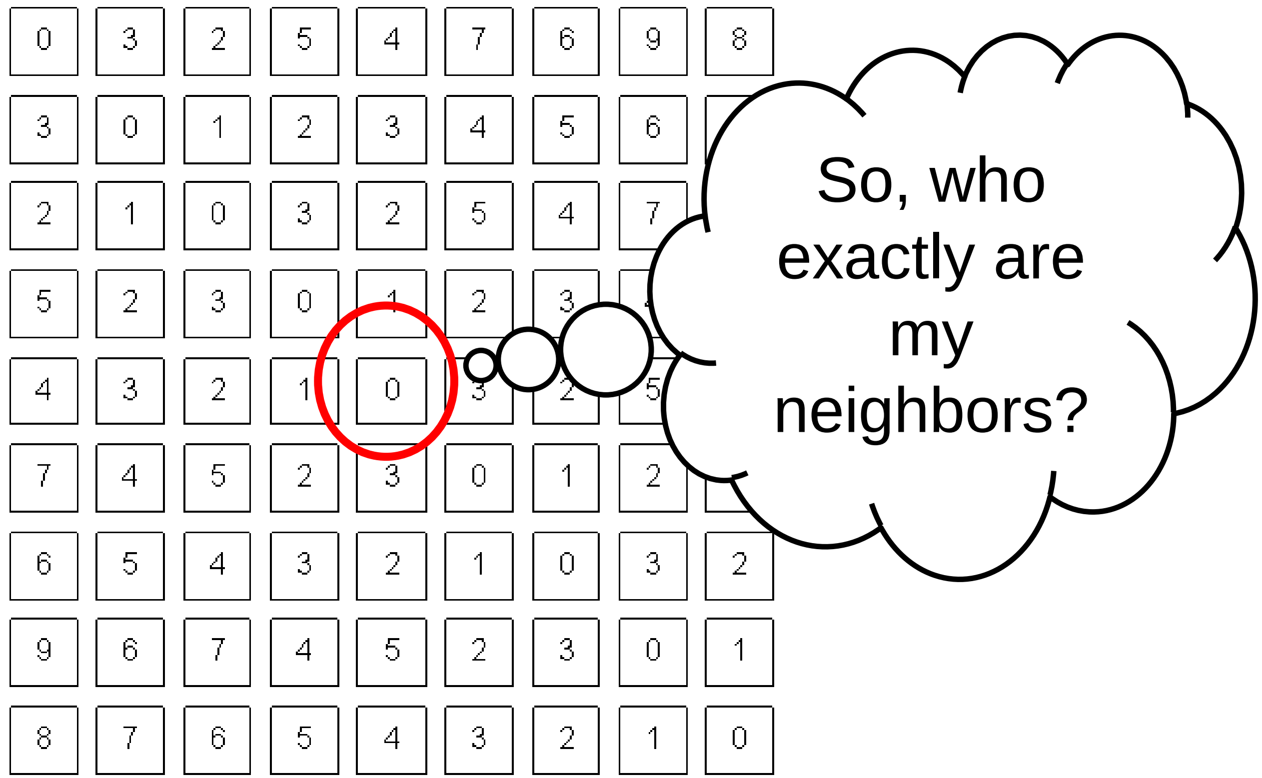
- Single Pixel Manipulation
  – Dynamic Range Manipulation
  – Neighborhoods and Connectivity
  – Image Arithmetic
  – Example: Background Subtraction
- Frequency Space
- Digital Filters

Neighbors

Why do we care at all?



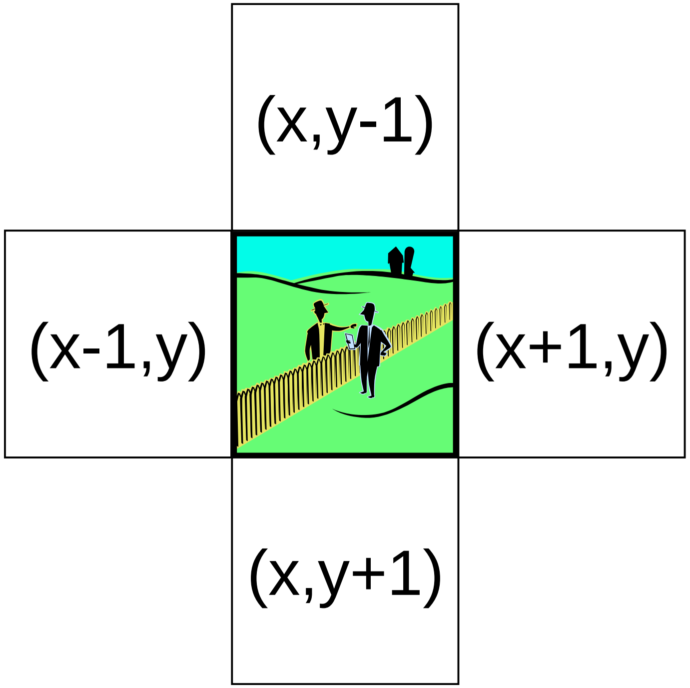
Digital Images



What a computer sees

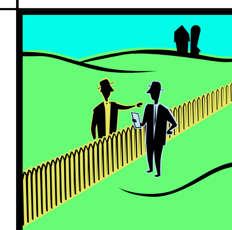
4-Neighbors

- A pixel p at (x,y) has 2 horizontal and 2 vertical neighbors:
  - (x+1,y), (x-1,y), (x,y+1), (x,y-1)
  - N_4(p): Set of the 4-neighbors of p.
- Limitations?



8-Neighbors

- A pixel has 4 diagonal neighbors
  - (x+1,y+1), (x+1,y-1), (x-1,y+1), (x-1,y-1)
  - $N_D(p)$: Diagonal set of neighbors
- $N_8(p) = N_4(p) + N_D(p)$
- Limitations?

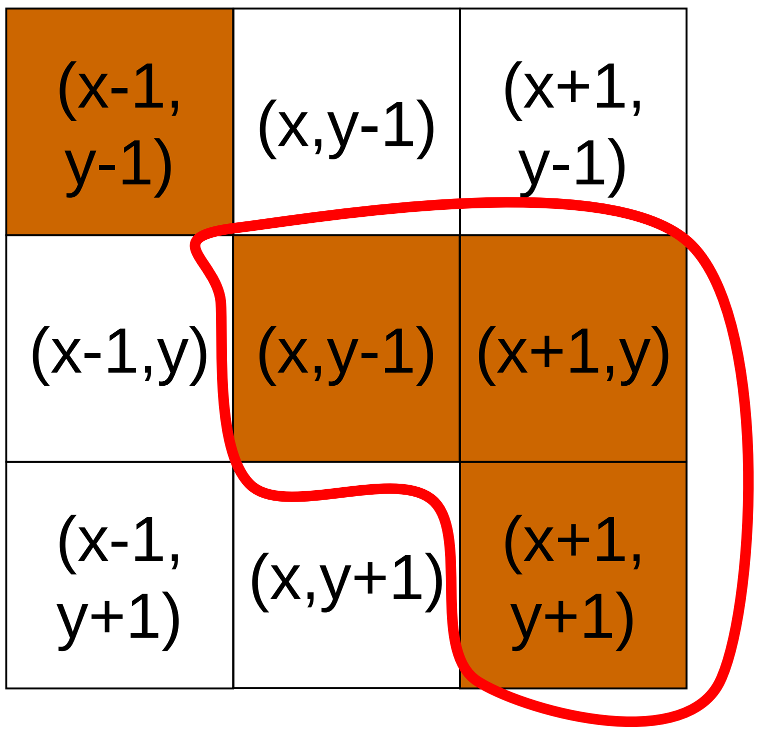
(x-1, y-1)	(x,y-1)	(x+1, y-1)
(x-1,y)		(x+1,y)
(x-1, y+1)	(x,y+1)	(x+1, y+1)

Connectivity

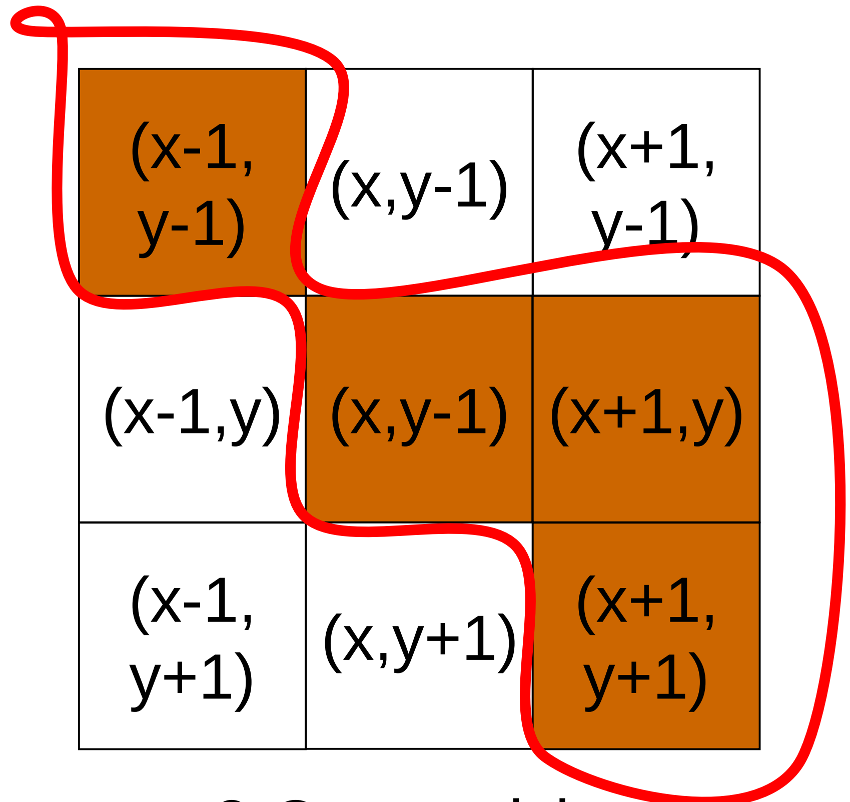
- Two pixels are connected if:
  - They are neighbors (i.e. adjacent in some sense -- e.g. $N_4(p)$, $N_8(p)$, …)
  - Their gray levels satisfy a specified criterion of similarity (e.g. equality, …)

(x-1, y-1)	(x,y-1)	(x+1, y-1)
(x-1,y)	(x,y-1)	(x+1,y)
(x-1, y+1)	(x,y+1)	(x+1, y+1)

4 and 8-Connectivity

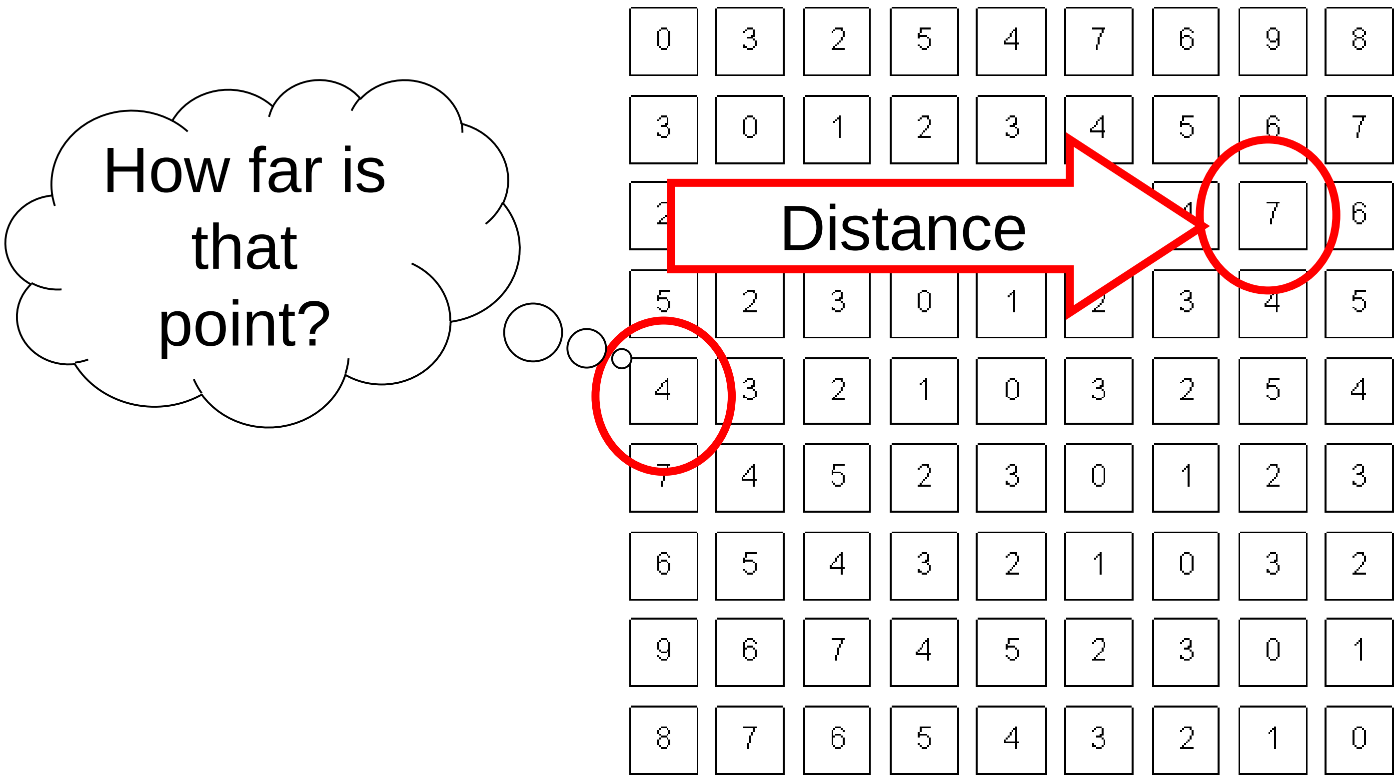


4-Connectivity



8-Connectivity

Distances



D4 Distance

- D_4 distance (city-block distance):
  - $D_4(p,q) = |x-s| + |y-t|$
  - forms a diamond centered at (x,y)
  - e.g. pixels with $D_4 \leq 2$ from p

		2		
	2	1	2	
2	1	0	1	2
	2	1	2	
		2		

$D_4 = 1$ are the 4-neighbors of p

D8 Distance

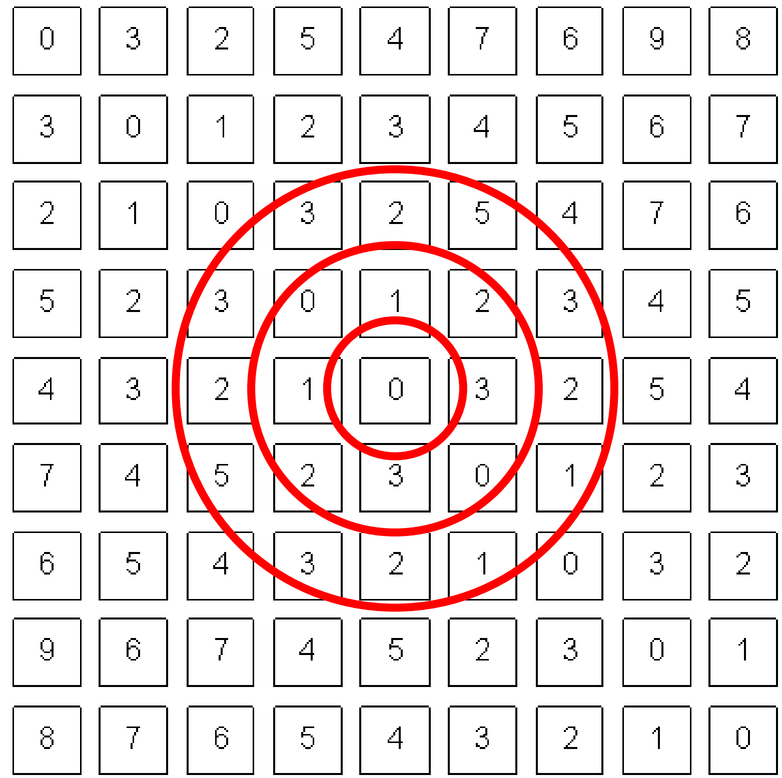
- D_8 distance (chessboard distance):
  - $D_8(p,q) = \max(|x-s|,|y-t|)$
  - Forms a square centered at p
  - e.g. pixels with $D_8 \leq 2$ from p

2	2	2	2	2
2	1	1	1	2
2	1	0	1	2
2	1	1	1	2
2	2	2	2	2

$D_8 = 1$ are the 8-neighbors of p

Euclidean Distance

- Euclidean distance:
  - $D_e(p,q) = [(x-s)^2 + (y-t)^2]^{1/2}$
  - Points (pixels) having a distance less than or equal to r from (x,y) are contained in a disk of radius r centered at (x,y).



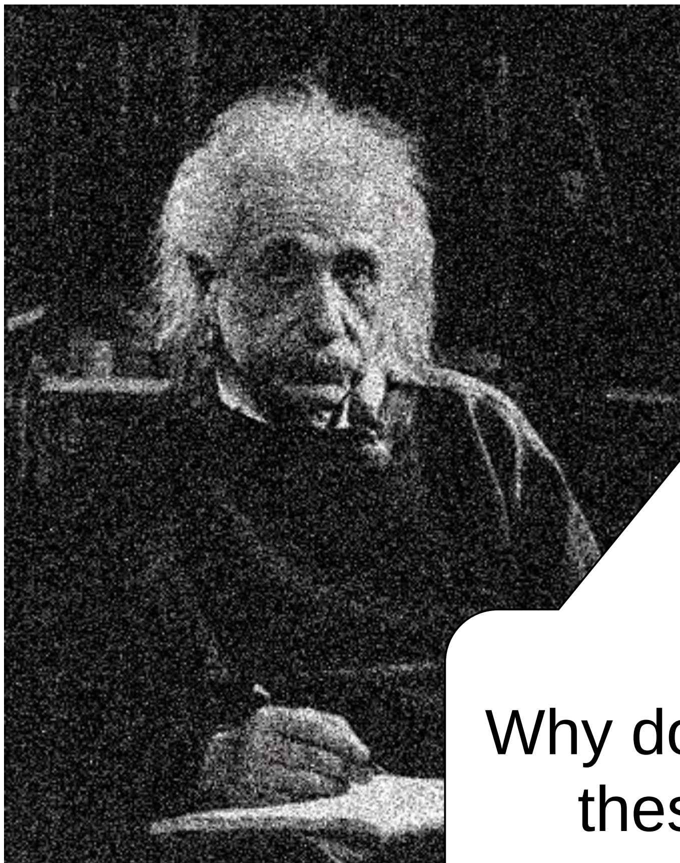
Outline

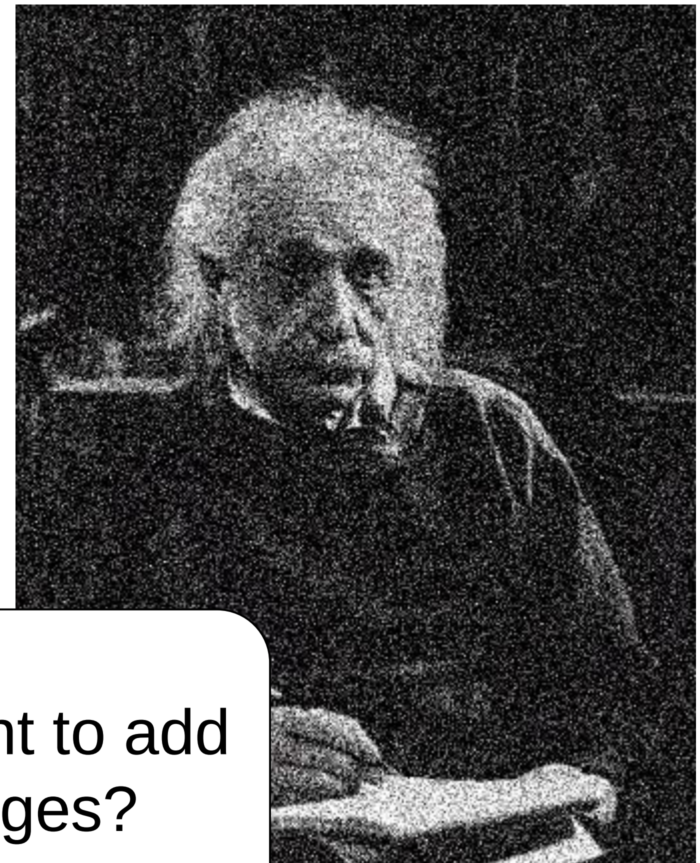
- Single Pixel Manipulation
  - Dynamic Range Manipulation
  - Neighborhoods and Connectivity
  - Image Arithmetic
  - Example: Background Subtraction
- Frequency Space
- Digital Filters

Arithmetic operations between images

Why do I want to add these images?

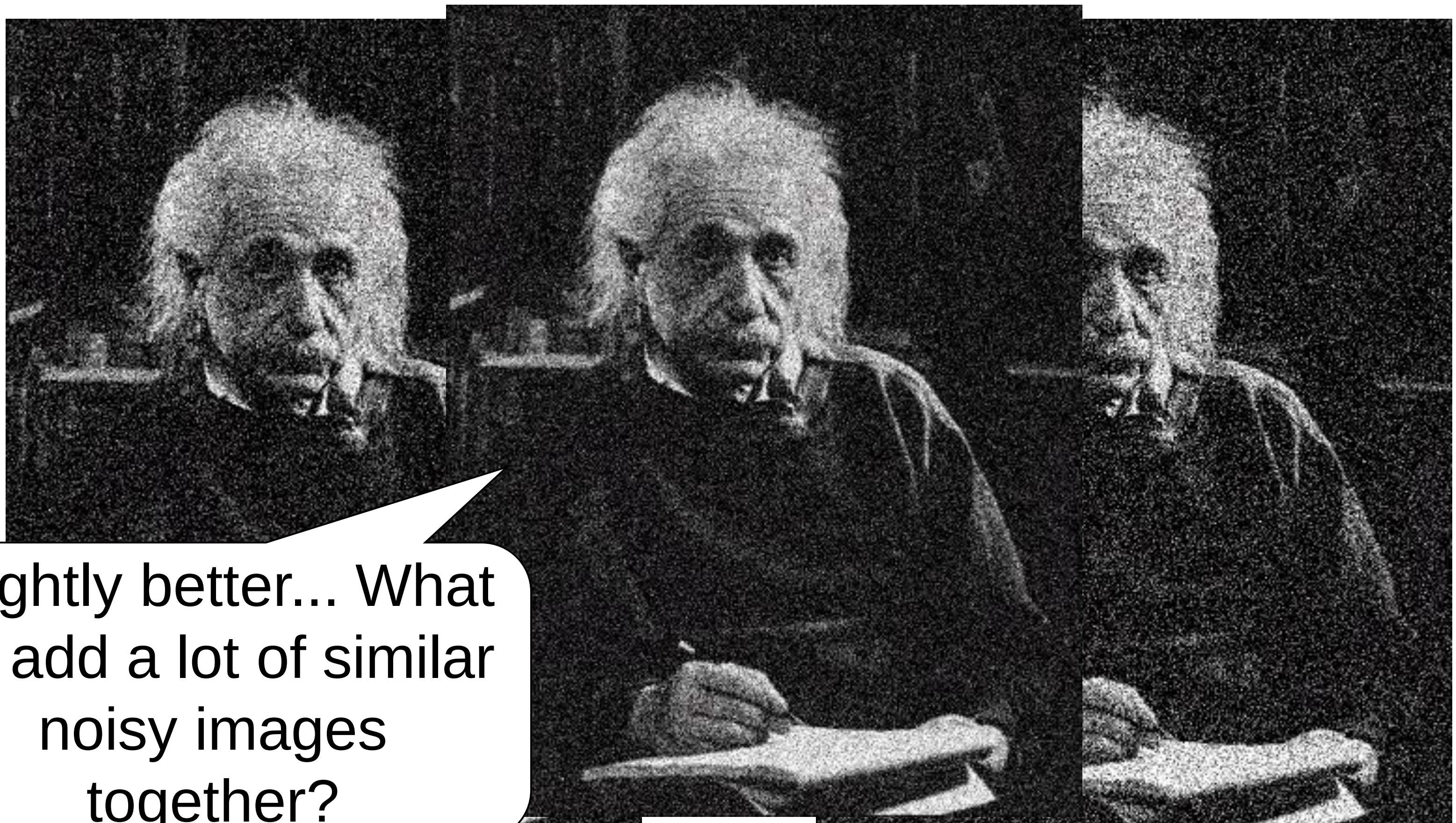
Arithmetic operations between images





Why do I want to add these images?

Arithmetic operations between images



Slightly better... What if I add a lot of similar noisy images together?

Image Arithmetic

- Image 1: a(x,y)
- Image 2: b(x,y)
- Result: c(x,y) = a(x,y) OPERATION b(x,y)
- Possibilities:
  - Addition
  - Subtraction
  - Multiplication
  - Division
  - Etc..

Why is this useful? What problems can happen?

Logic Operations

- Binary Images
- We can use Boolean Logic
- Operations:
  - AND
  - OR
  - NOT

More on this when we study mathematical morphology.

Outline

- Single Pixel Manipulation
  - Dynamic Range Manipulation
  - Neighborhoods and Connectivity
  - Image Arithmetic
  - Example: Background Subtraction
- Frequency Space
- Digital Filters

Example: Background Subtraction

- Image arithmetic is simple and powerful.

Is there a person here? Where?

Background Subtraction

- Remember: We can only see numbers!

Is there a person here? Where?

0	3	2	5	4	7	6	9	8
3	0	1	2	3	4	5	6	7
2	1	0	3	2	5	4	7	6
[illegible]	2	3	0	1	2	3	4	5
4	3	2	1	0	3	2	5	4
7	4	5	2	3	0	1	2	3
6	5	4	3	2	1	0	3	2
9	6	7	4	5	2	3	0	1
8	7	6	5	4	3	2	1	0

Background Subtraction

- What if I know this?

Background Subtraction

- Subtract!
- Limitations?

Background Subtraction

- Objective:
  - Separate the foreground objects from a static background.
- Large variety of methods:
  - Mean & Threshold [CD04]
  - Normalized Block Correlation [Mats00]
  - Temporal Derivative [Hari98]
  - Single Gaussian [Wren97]
  - Mixture of Gaussians [Grim98]

Segmentation!! More on this later.

Outline

- Single Pixel Manipulation
- Frequency Space
  - Fourier Transform
  - Frequency Space
  - Spatial Convolution
- Digital Filters

Outline

- Single Pixel Manipulation
- Frequency Space
  - Fourier Transform
  - Frequency Space
  - Spatial Convolution
- Digital Filters

How to Represent Signals?

- Option 1: Taylor series represents any function using polynomials.

$$f(x) = f(a) + f'(a)(x-a) + \frac{f''(a)}{2!}(x-a)^2 + \frac{f^{(3)}(a)}{3!}(x-a)^3 + \ldots + \frac{f^{(n)}(a)}{n!}(x-a)^n + \ldots$$

- Polynomials are not the best - unstable and not very physically meaningful.

- Easier to talk about “signals” in terms of its “frequencies” (how fast/often signals change, etc).

Jean Baptiste Joseph Fourier (1768-1830)

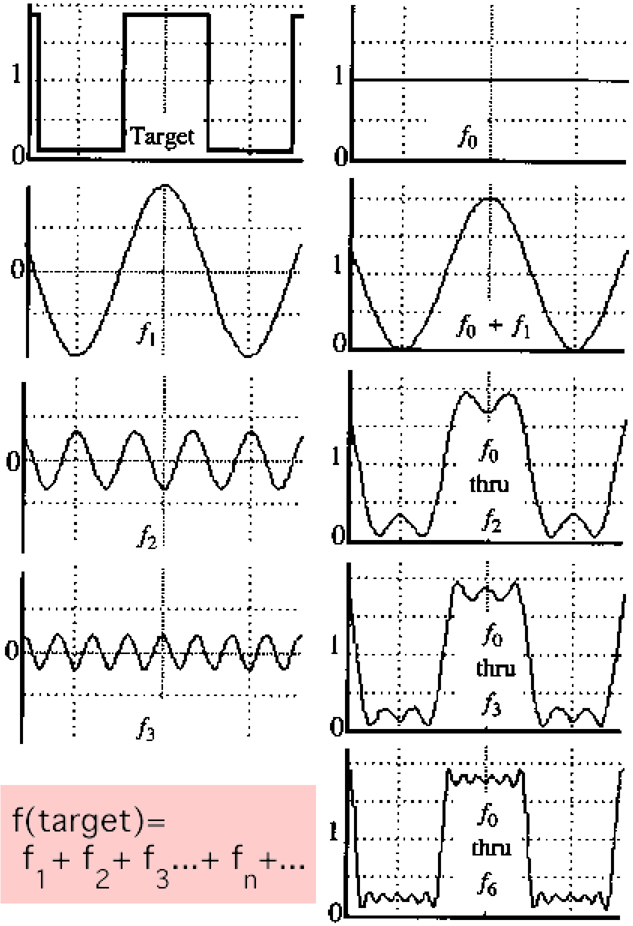
- Had a crazy idea (1807):
- **Any** periodic function can be rewritten as a weighted sum of Sines and Cosines of different frequencies.
- Don't believe it?
  - Neither did Lagrange, Laplace, Poisson and other big wigs
  - Not translated into English until 1878!
- But it's true!
  - called Fourier Series
  - Possibly the greatest tool used in Engineering

A Sum of Sinusoids

- Our building block:

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

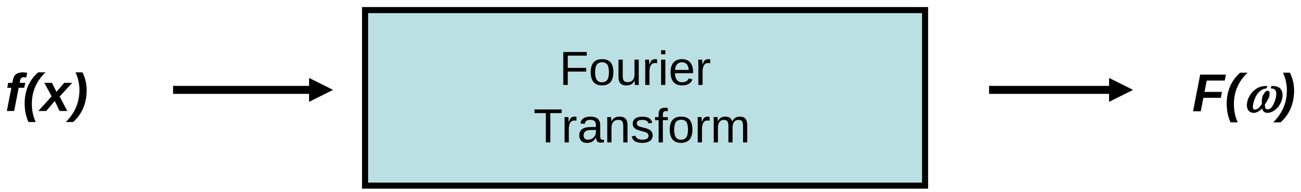
- Add enough of them to get any signal *f(x)* you want!
- How many degrees of freedom?
- What does each control?
- Which one encodes the coarse vs. fine structure of the signal?



f(target)=
$f_1 + f_2 + f_3 ... + f_n + ...$

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
  - How can *F* hold both? Complex number trick!

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

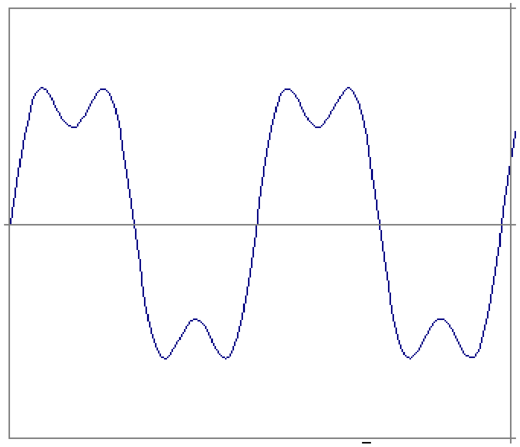
$$F(\omega) = R(\omega) + iI(\omega)$$

$$A = \pm\sqrt{R(\omega)^2 + I(\omega)^2}$$

$$\phi = \tan^{-1}\frac{I(\omega)}{R(\omega)}$$

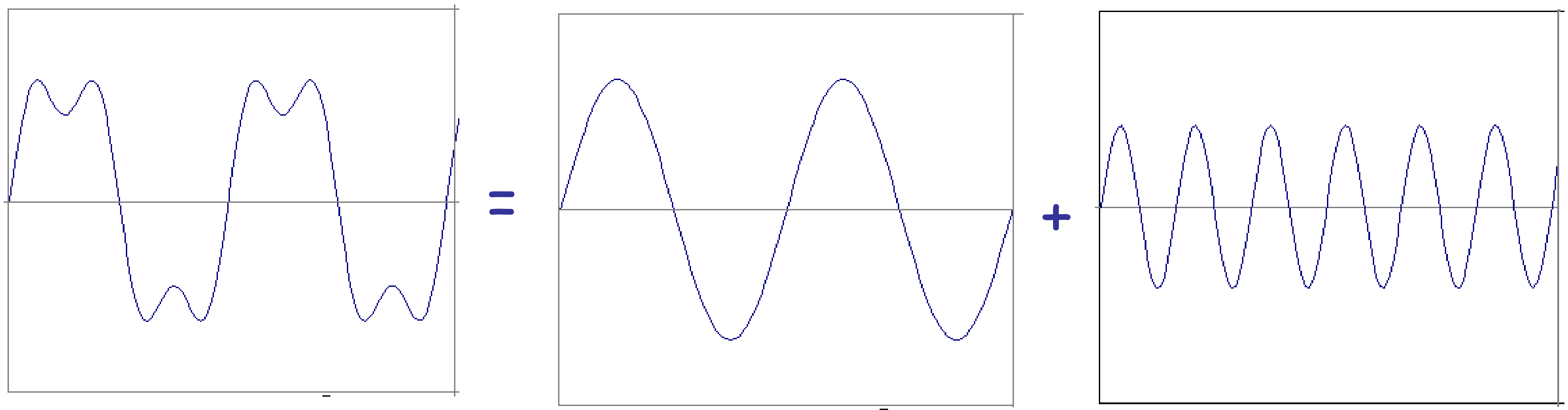
Time and Frequency

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



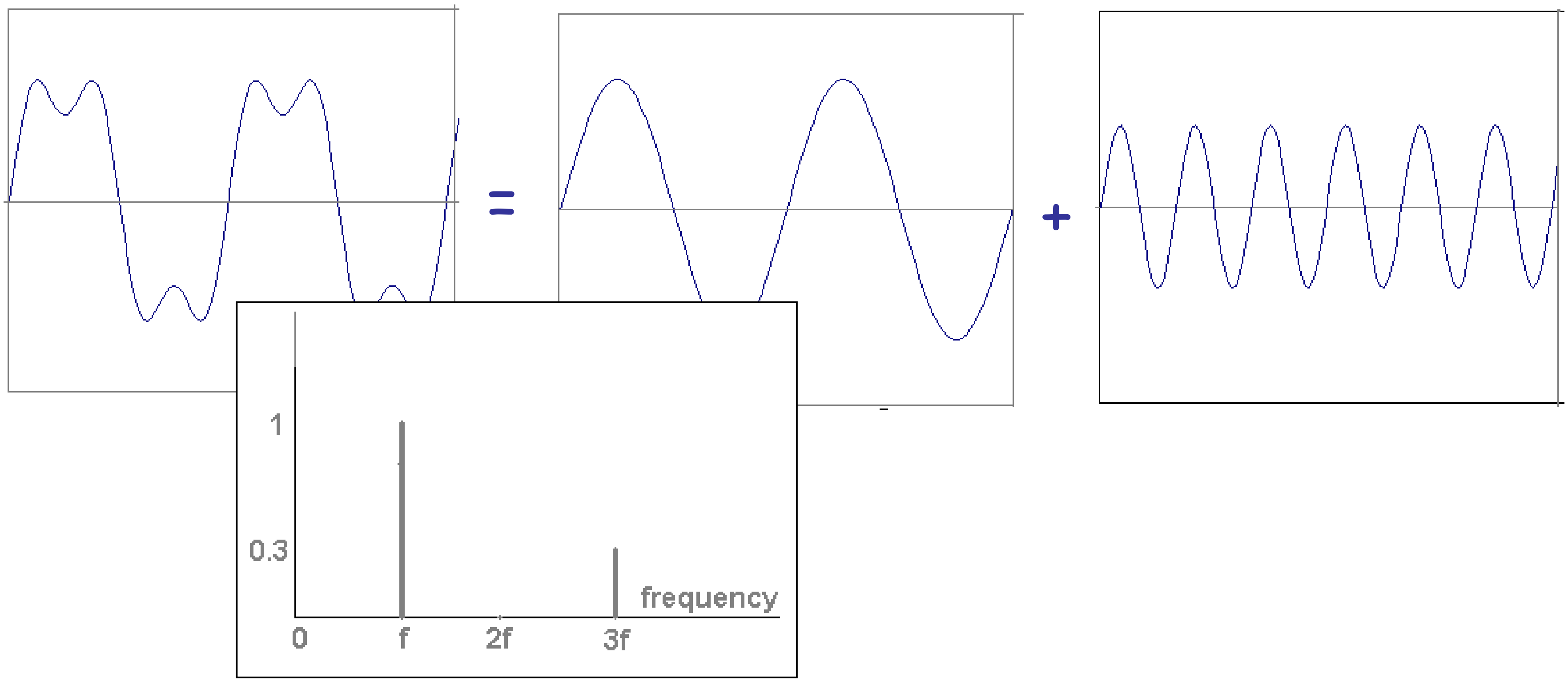
Time and Frequency

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



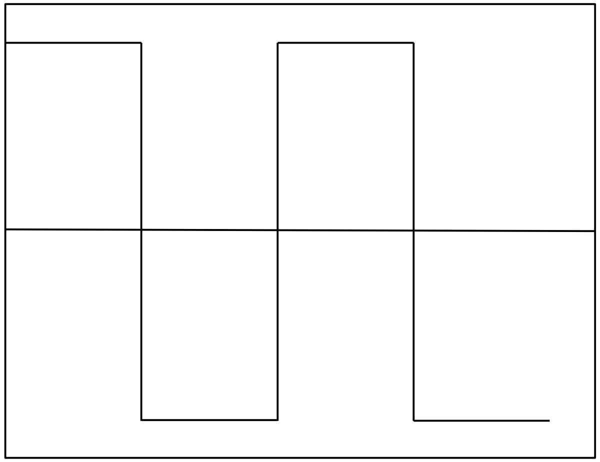
Frequency Spectra

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

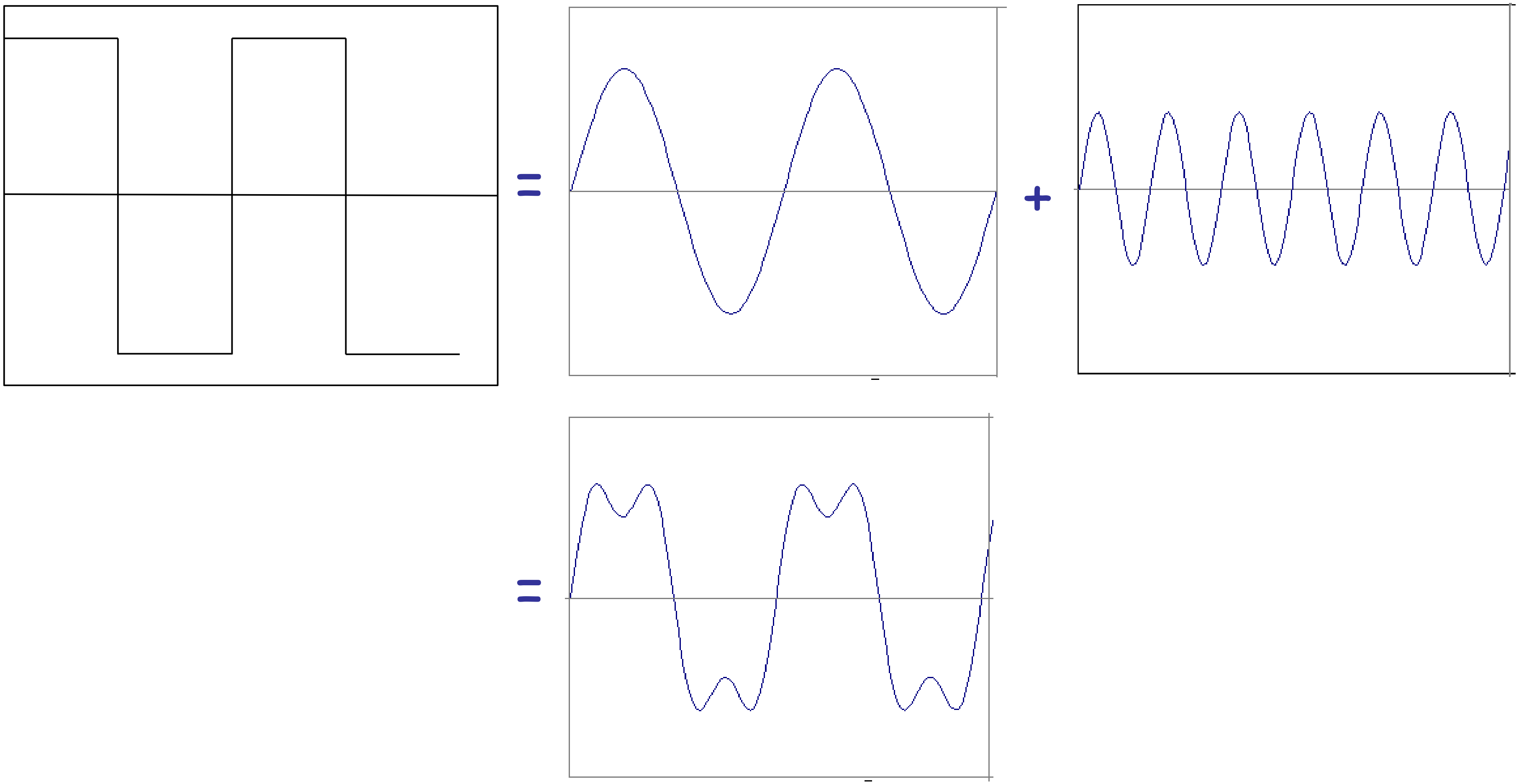


Frequency Spectra

- Usually, frequency is more interesting than the phase

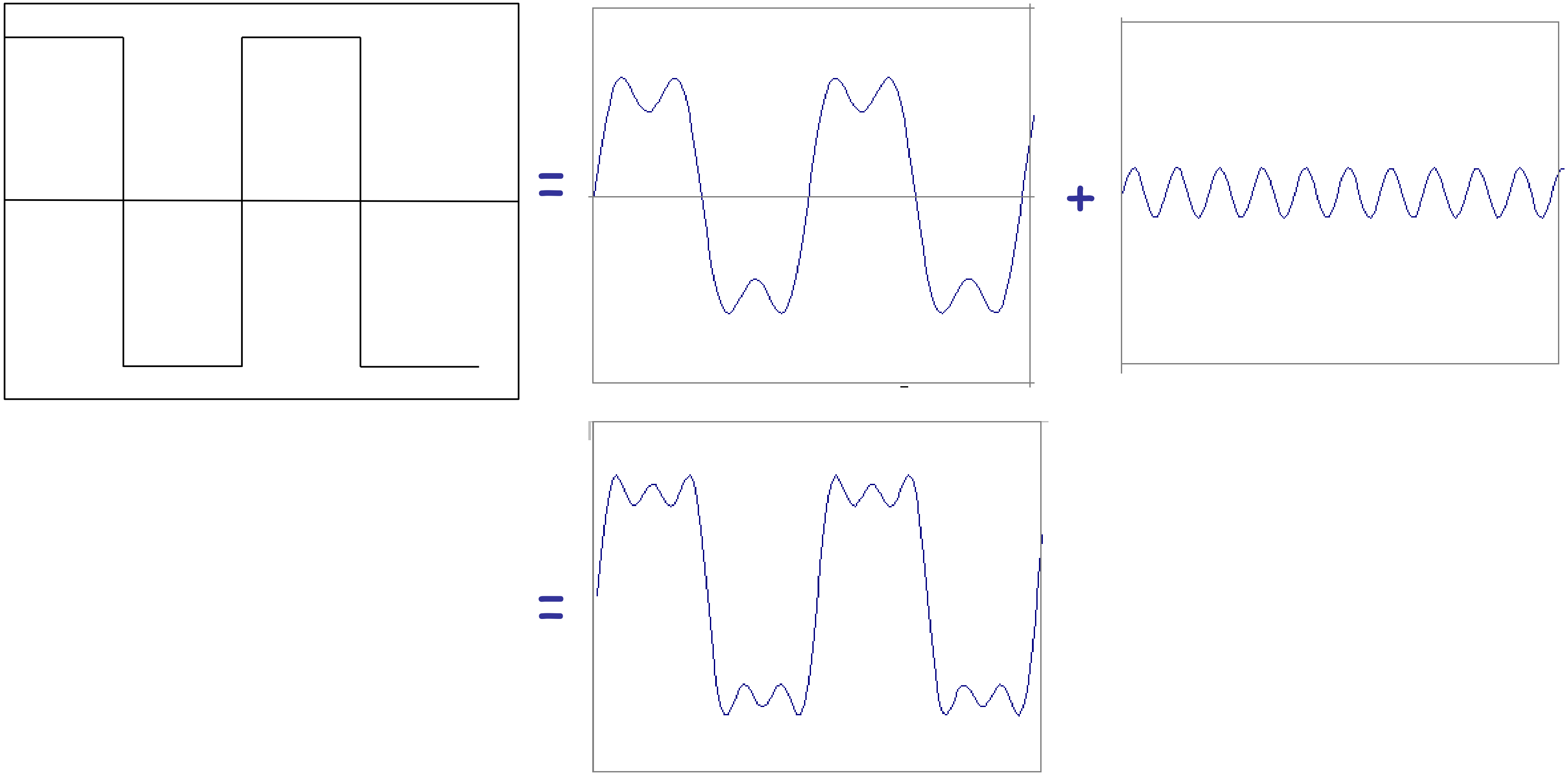


Frequency Spectra

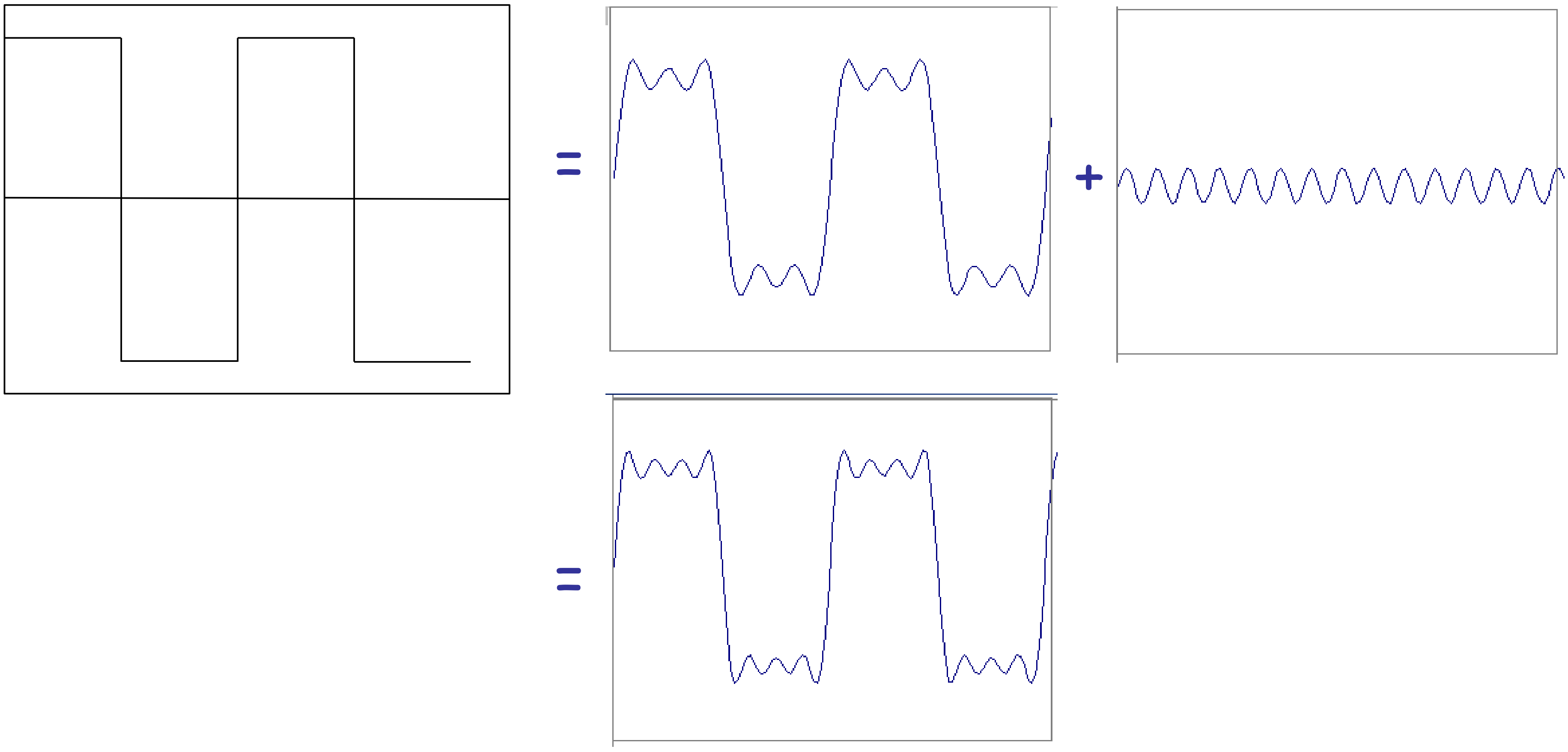


DOCTORAL PROGRAMME
IN COMPUTER SCIENCE

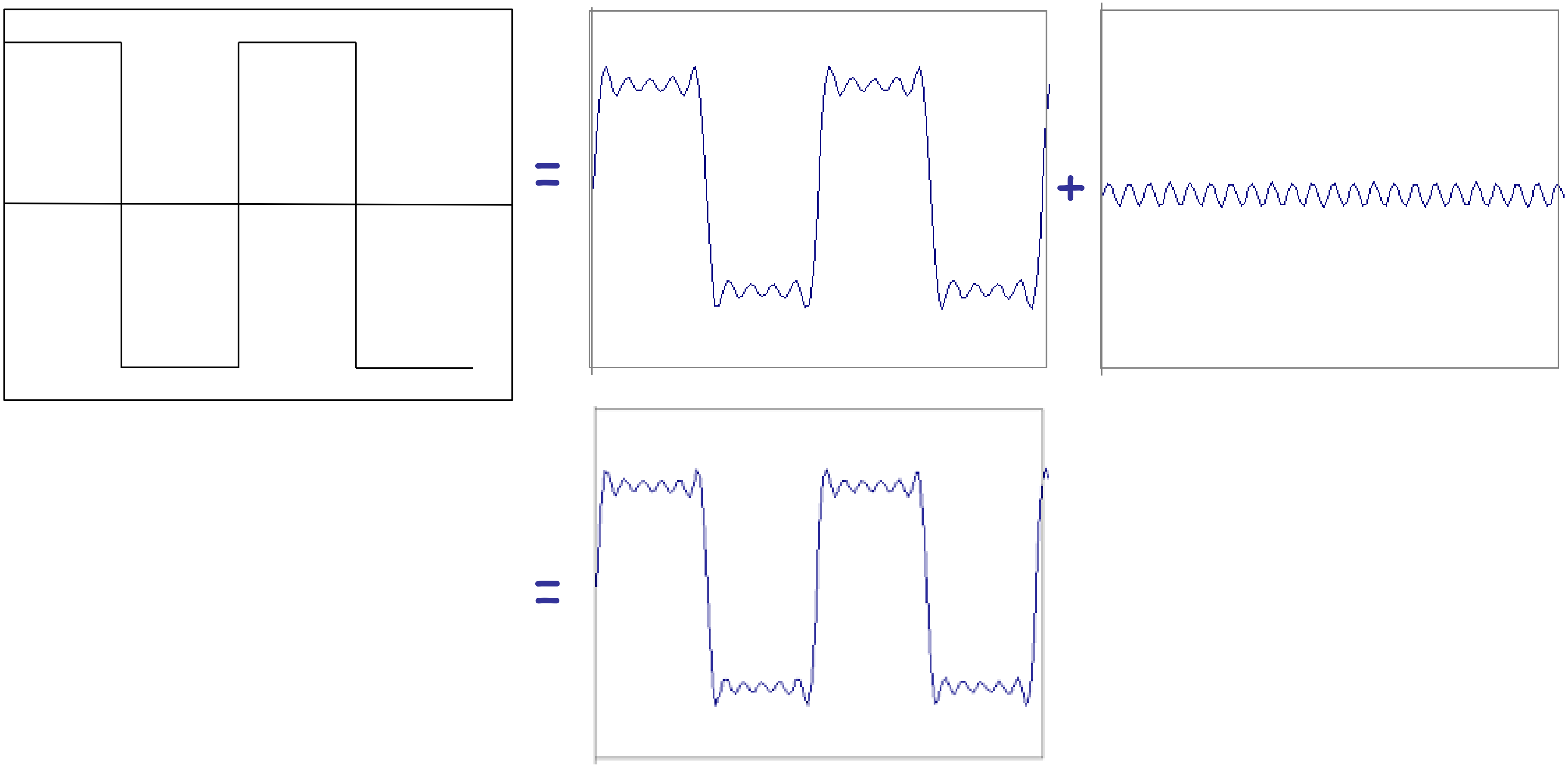
Frequency Spectra



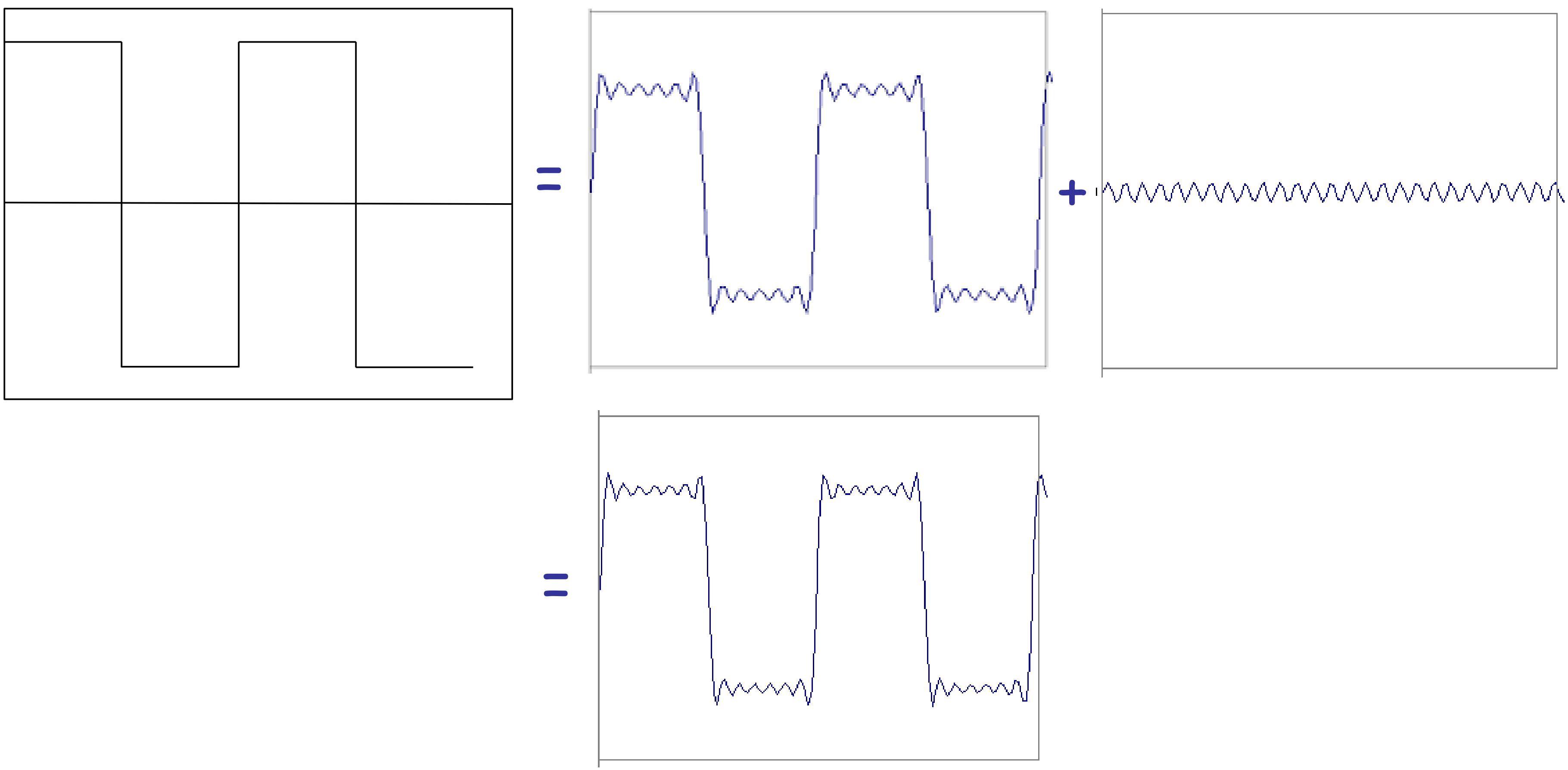
Frequency Spectra



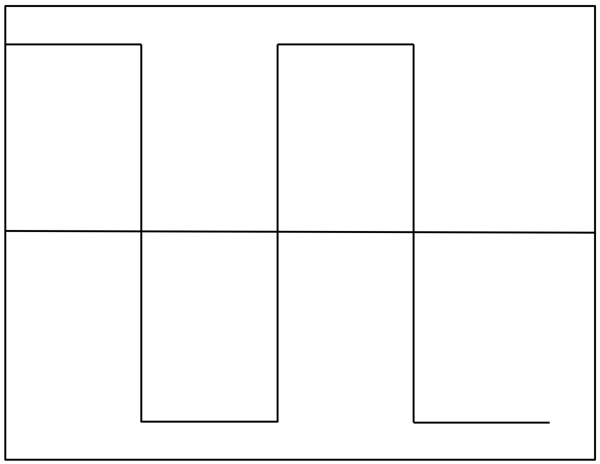
Frequency Spectra



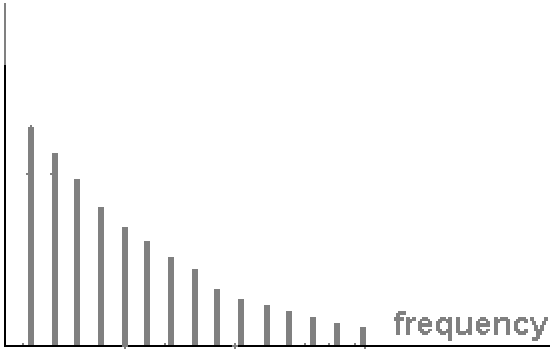
Frequency Spectra



Frequency Spectra



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



Fourier Transform – more formally

Represent the signal as an infinite weighted sum of an infinite number of sinusoids

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

Note: $e^{ik}=\cos k + i\sin k \qquad i=\sqrt{-1}$

Arbitrary function ⟶ Single Analytic Expression

Spatial Domain (*x*) ⟶ Frequency Domain (*u*)

(Frequency Spectrum *F(u)*)

Inverse Fourier Transform (IFT) $$f(x)=\int_{-\infty}^{\infty} F(u)e^{i2\pi ux}dx$$

Properties of Fourier Transform

Property	Spatial Domain	Frequency Domain
Linearity	$c_1 f(x) + c_2 g(x)$	$c_1 F(u) + c_2 G(u)$
Scaling	$f(ax)$	$\frac{1}{\lvert a \rvert} F\left(\frac{u}{a}\right)$
Shifting	$f(x - x_0)$	$e^{-i2\pi u x_0} F(u)$
Symmetry	$F(x)$	$f(-u)$
Conjugation	$f^*(x)$	$F^*(-u)$
Convolution	$f(x) * g(x)$	$F(u)G(u)$
Differentiation	$\frac{d^n f(x)}{dx^n}$	$(i2\pi u)^n F(u)$

Outline

- Single Pixel Manipulation
- Frequency Space
  - Fourier Transform
  - Frequency Space
  - Spatial Convolution
- Digital Filters

How does this apply to images?

- We have defined the Fourier Transform as

$$F(u)=\int_{-\infty}^{\infty} f(x)e^{-iux}dx$$

- But images are:
  - Discrete.
  - Two-dimensional.

0	3	2	5	4	7	6	9	8
3	0	1	2	3	4	5	6	7
2	1	0	3	2	5	4	7	6
5	2	3	0	1	2	3	4	5
4	3	2	1	0	3	2	5	4
7	4	5	2	3	0	1	2	3
6	5	4	3	2	1	0	3	2
9	6	7	4	5	2	3	0	1
8	7	6	5	4	3	2	1	0

What a computer sees

2D Discrete FT

- In a 2-variable case, the discrete FT pair is:

$$F(u,v) = \frac{1}{MN} \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f(x,y) \exp[-j2\pi(ux/M + vy/N)]$$

For u=0,1,2,…,M-1 and v=0,1,2,…,N-1

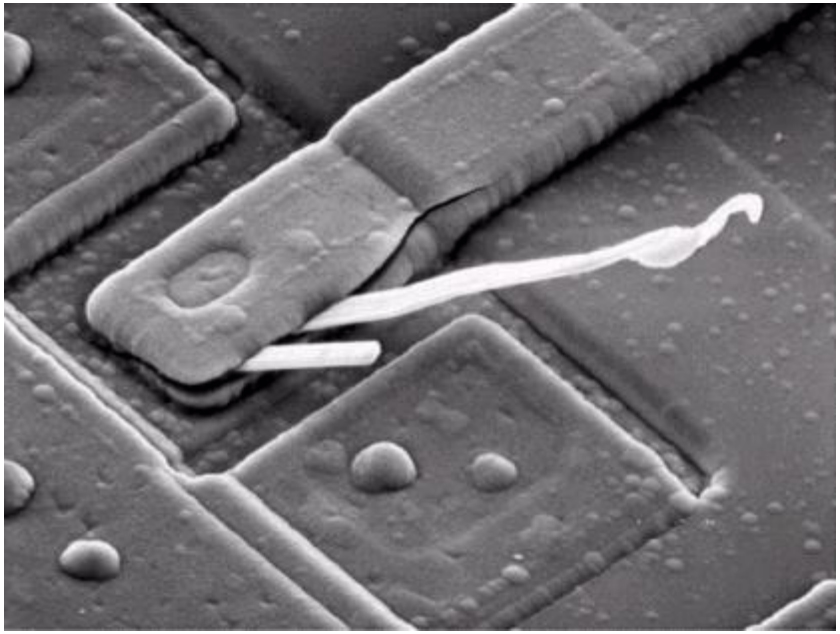
New matrix with the same size!

AND: $$f(x,y) = \sum_{u=0}^{M-1} \sum_{v=0}^{N-1} F(u,v) \exp[j2\pi(ux/M + vy/N)]$$

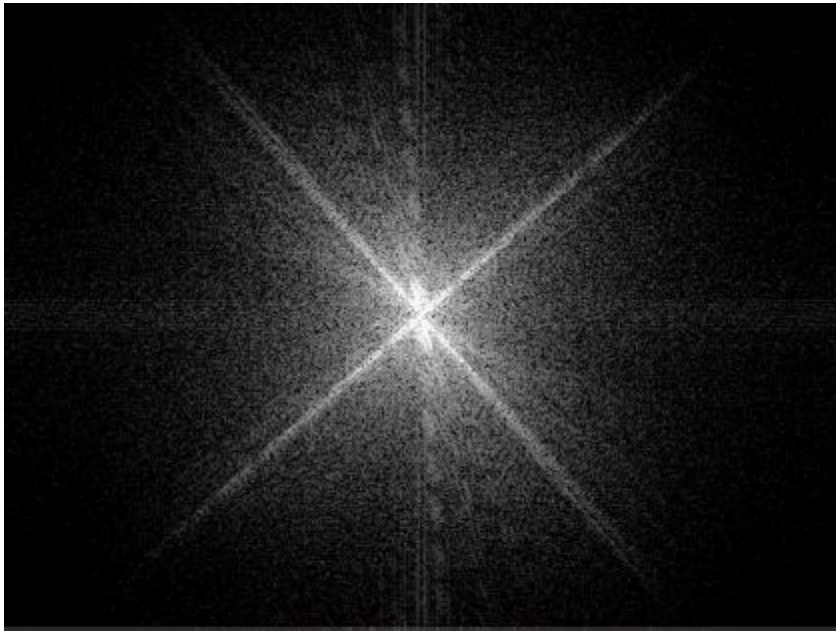
For x=0,1,2,…,M-1 and y=0,1,2,…,N-1

Frequency Space

- Image Space
  - f(x,y)
  - Intuitive

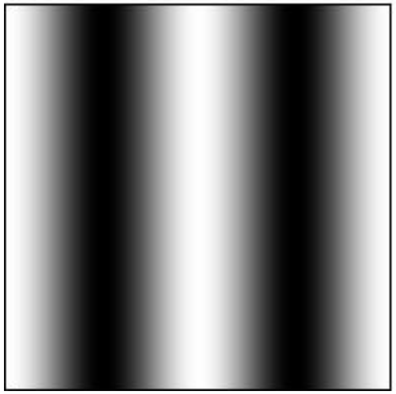


- Frequency Space
  - F(u,v)
  - What does this mean?



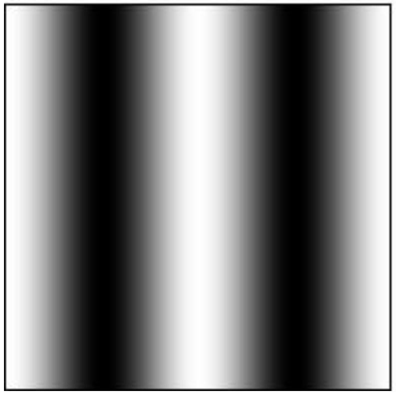
Frequency Space

- Basic Principles
  - The sinusoidal pattern shown below can be captured in a single Fourier term that encodes 1: the spatial frequency, 2: the magnitude (positive or negative), and 3: the phase.



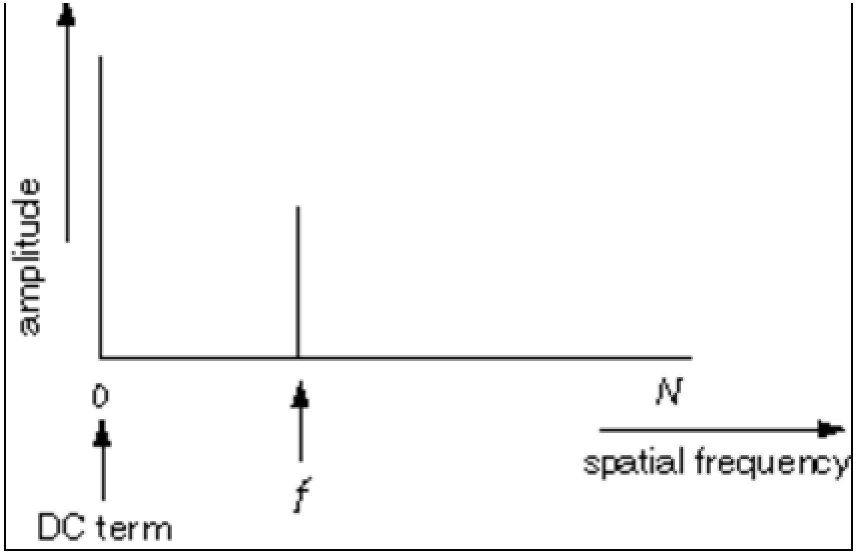
Frequency Space

- Basic Principles
  - The spatial frequency is the frequency across space (the x-axis in this case) with which the brightness modulates.
  - The magnitude of the sinusoid corresponds to its contrast, or the difference between the darkest and brightest peaks of the image. A negative magnitude represents a contrast-reversal, i.e. the brights become dark, and vice-versa.
  - The phase represents how the wave is shifted relative to the origin, in this case it represents how much the sinusoid is shifted left or right.



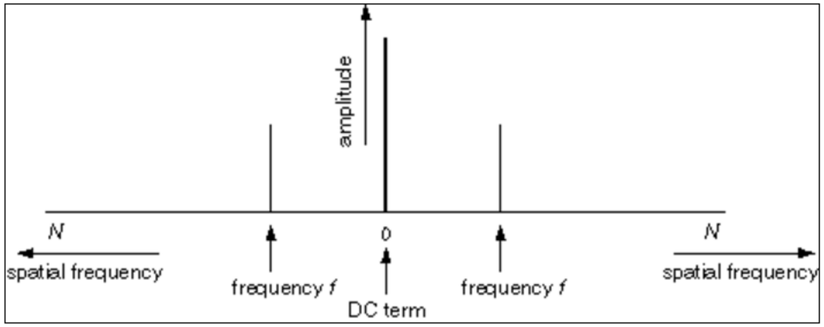
Frequency Space

- Basic Principles
  - The Fourier transform encodes all of the spatial frequencies present in an image simultaneously as follows. A signal containing only a single spatial frequency of frequency f is plotted as a single peak at point f along the spatial frequency axis, the height of that peak corresponding to the amplitude, or contrast of that sinusoidal signal.



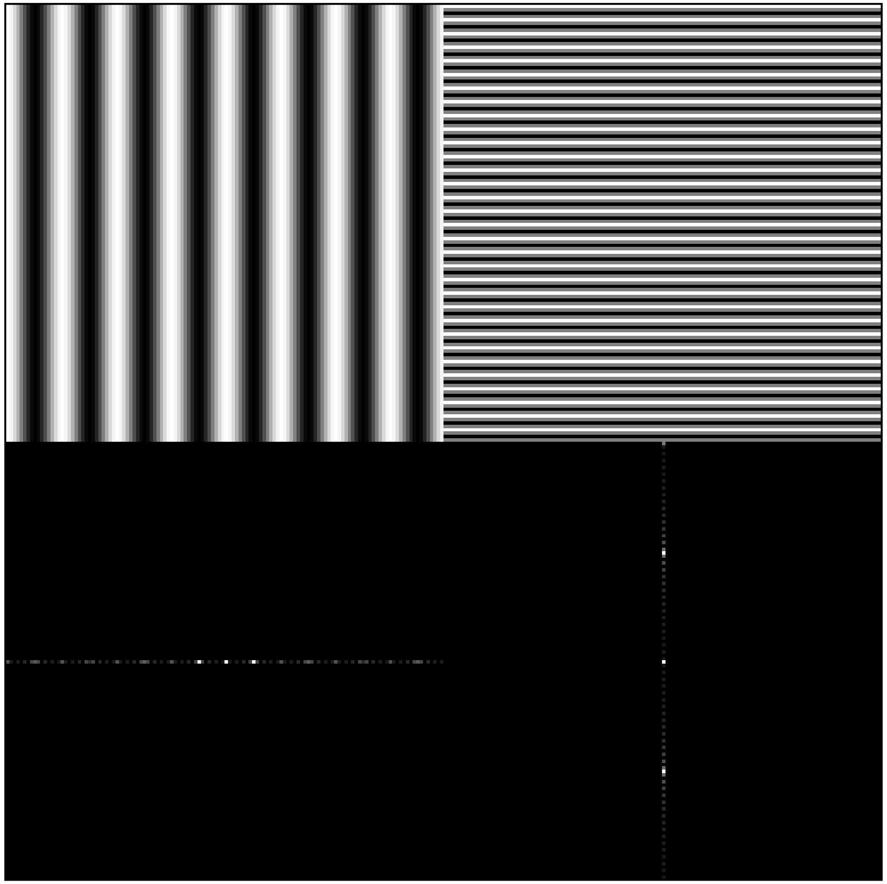
Frequency Space

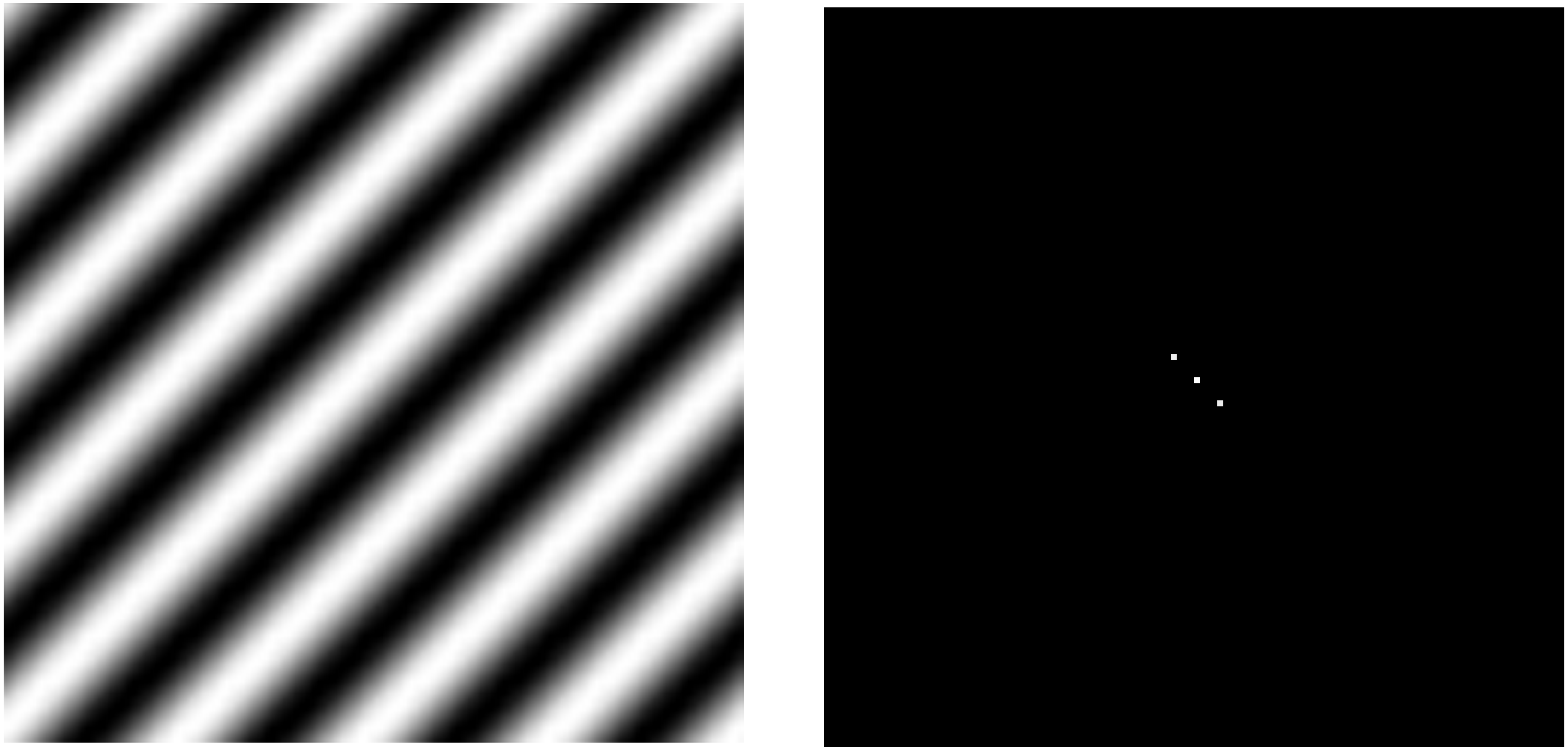
- Basic Principles
  - There is also a "DC term" corresponding to zero frequency, that represents the average brightness across the whole image. A zero DC term would mean an image with average brightness of zero, which would mean the sinusoid alternated between positive and negative values in the brightness image. But since there is no such thing as a negative brightness, all real images have a positive DC term.
  - Actually, for mathematical reasons beyond the scope of this tutorial, the Fourier transform also plots a mirror-image of the spatial frequency plot reflected across the origin, with spatial frequency increasing in both directions from the origin. For mathematical reasons beyond the scope of this explanation, these two plots are always mirror-image reflections of each other, with identical peaks at f and -f as shown below.

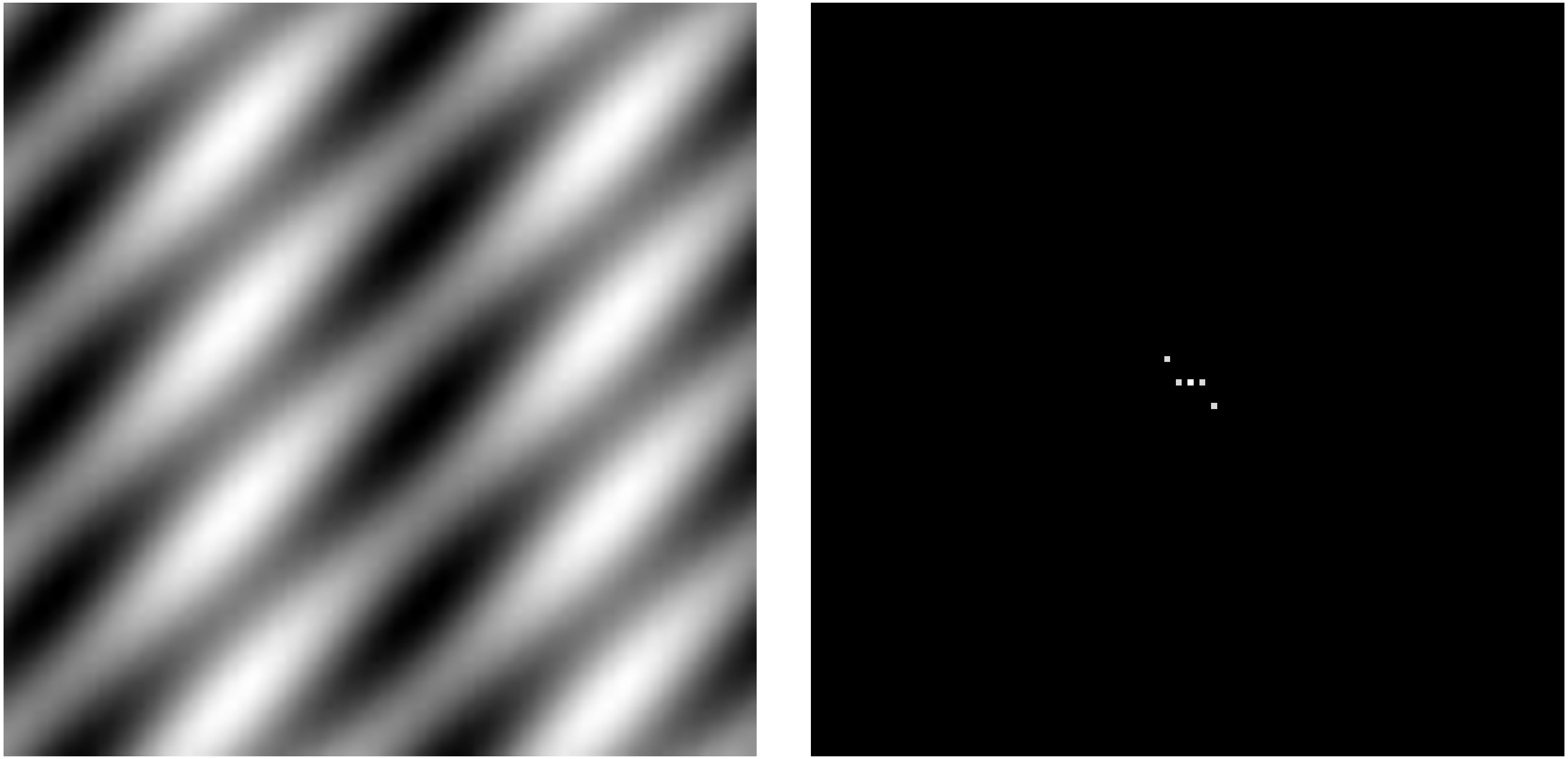


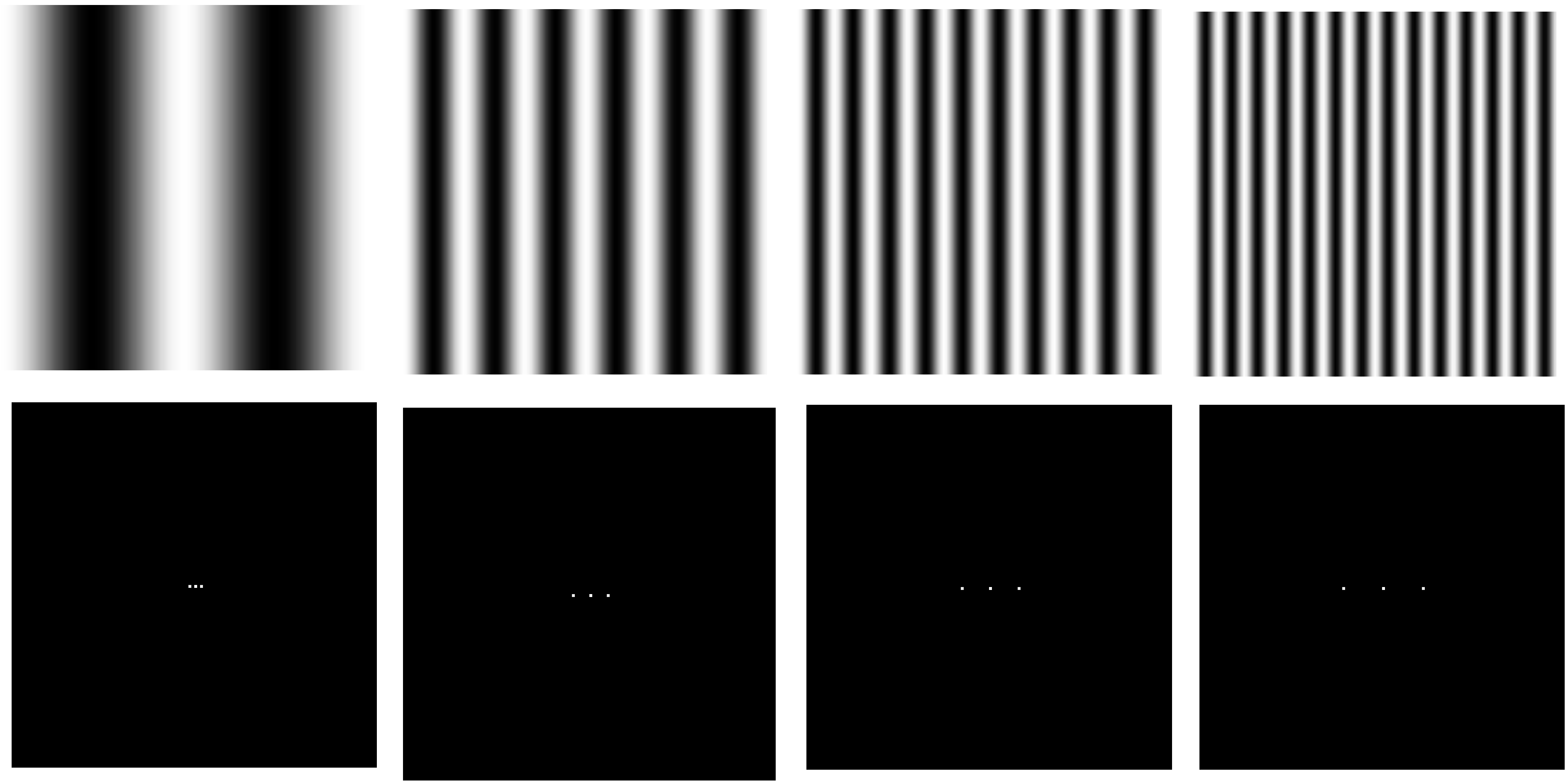
Horizontal and Vertical Frequency

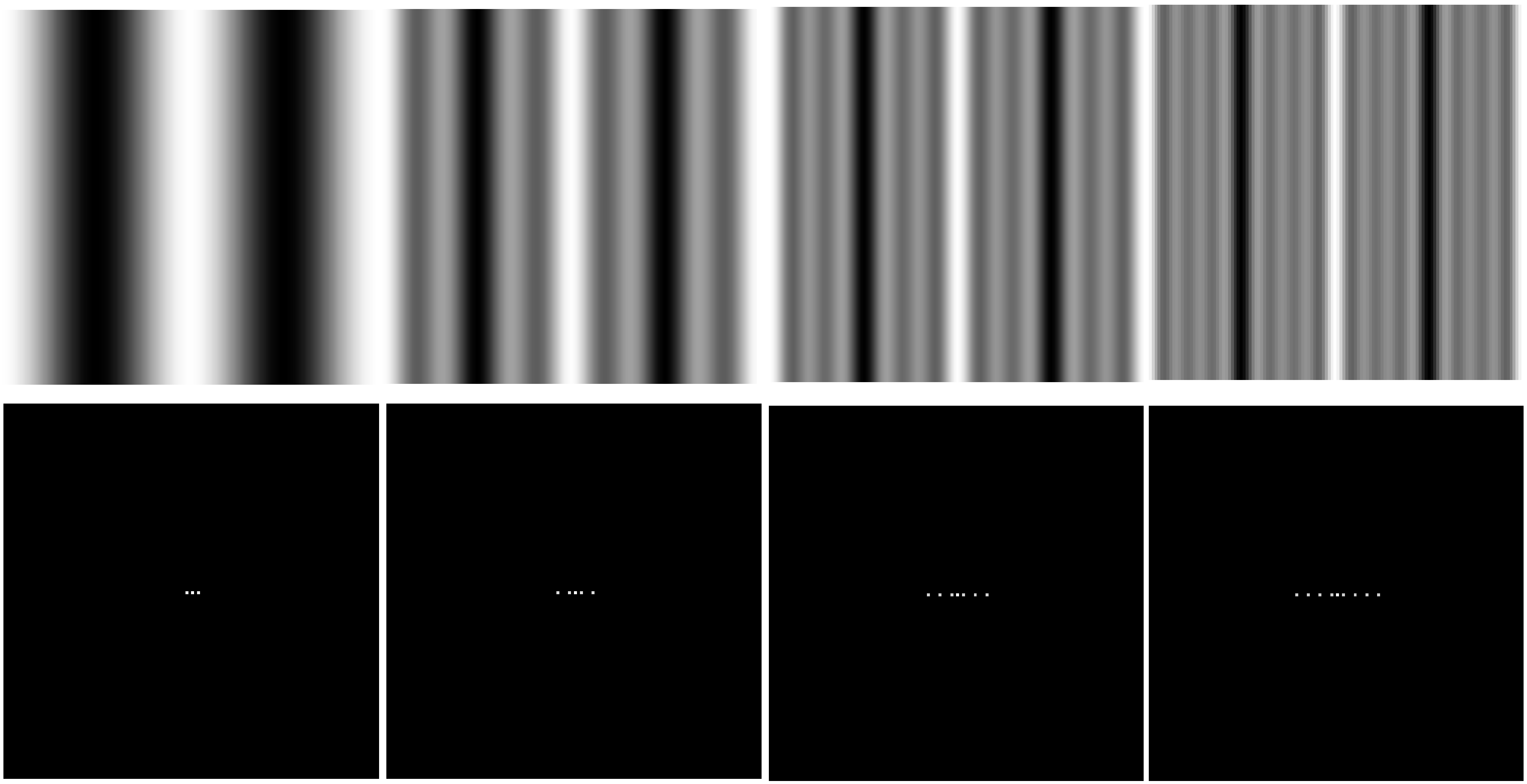
- Frequencies:
  - Horizontal frequencies correspond to horizontal gradients.
  - Vertical frequencies correspond to vertical gradients.
- The brighter the peaks in the Fourier image, the higher the contrast in the brightness image.
- What about diagonal lines?



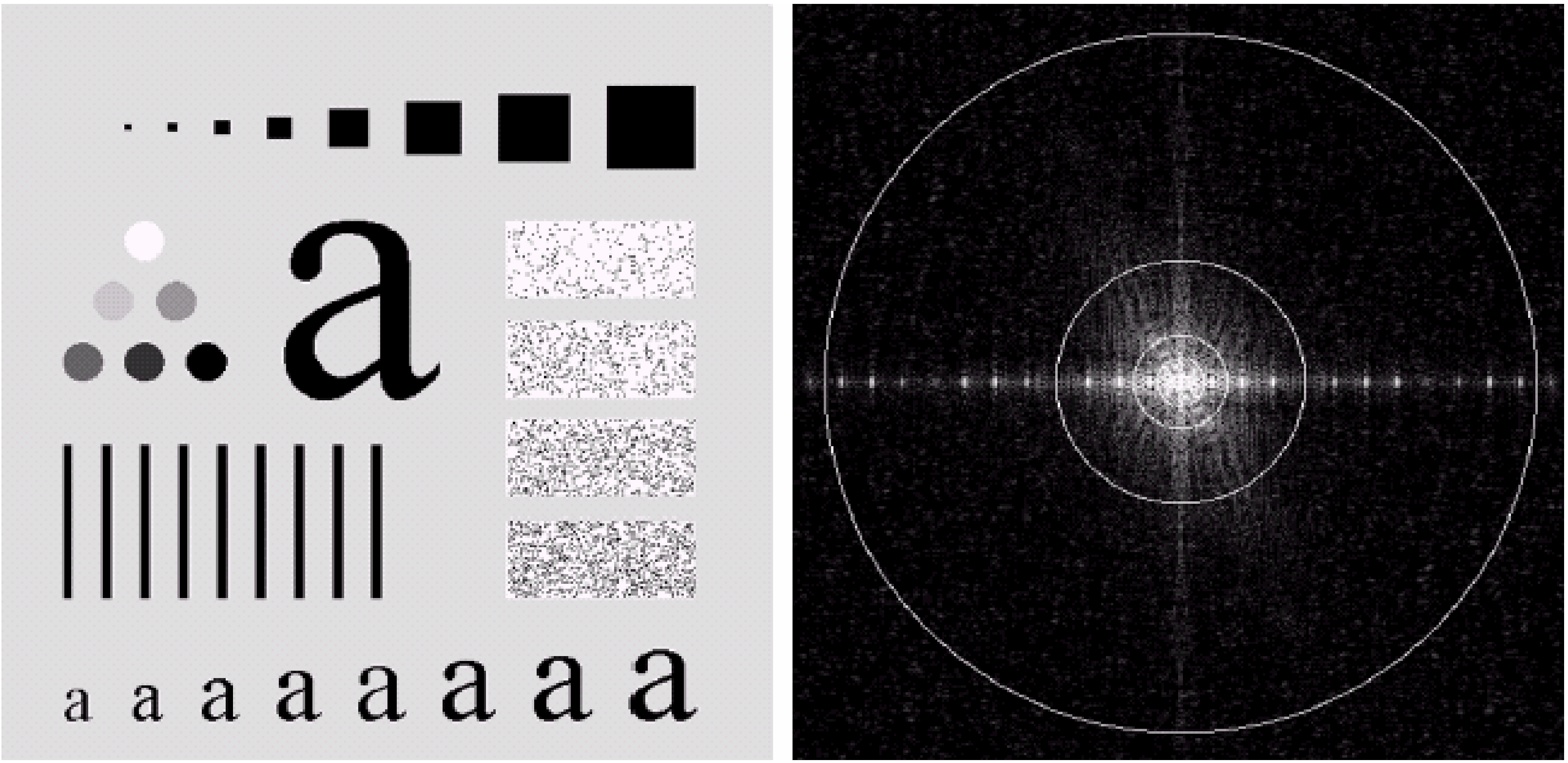




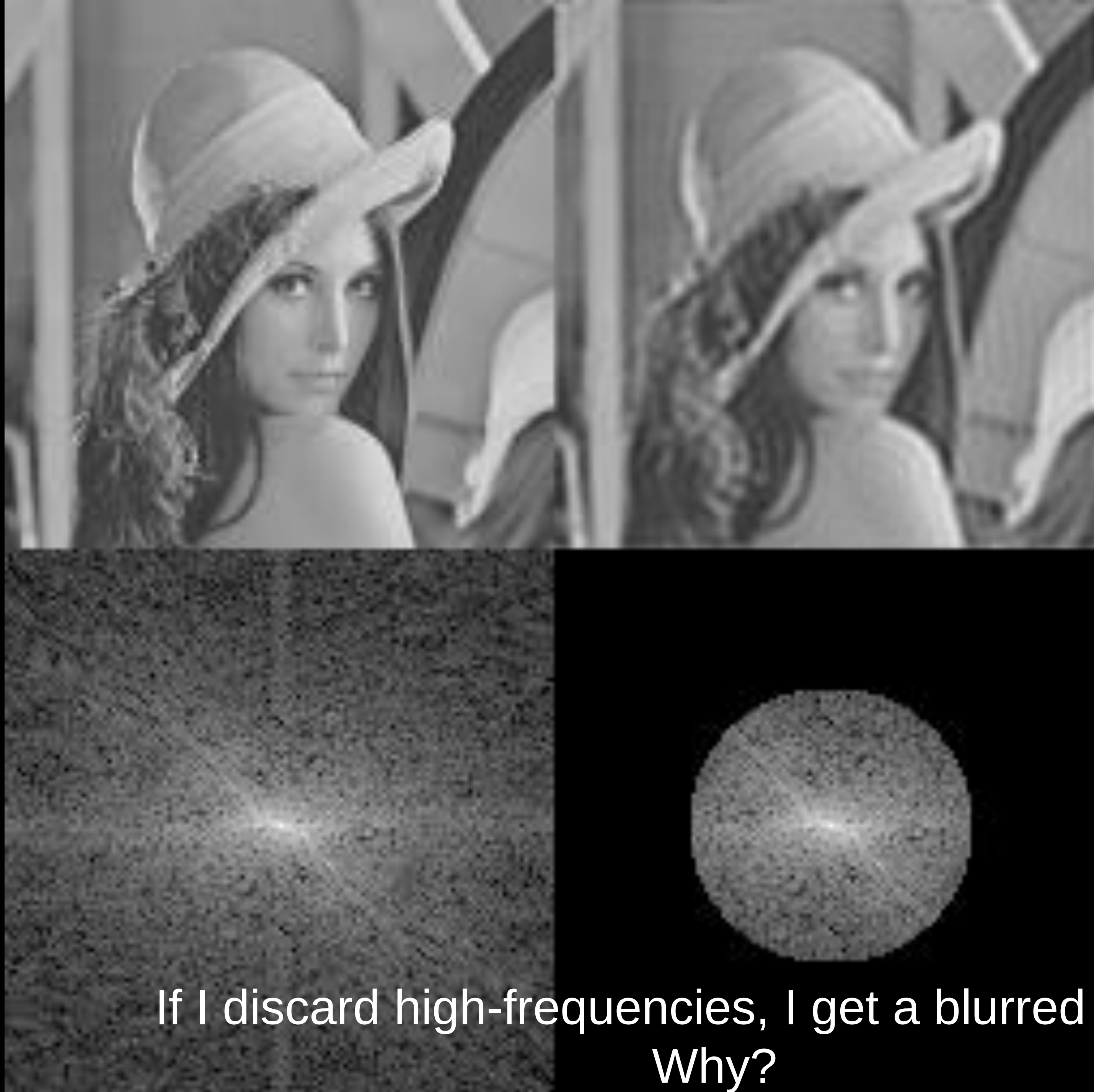




Power distribution

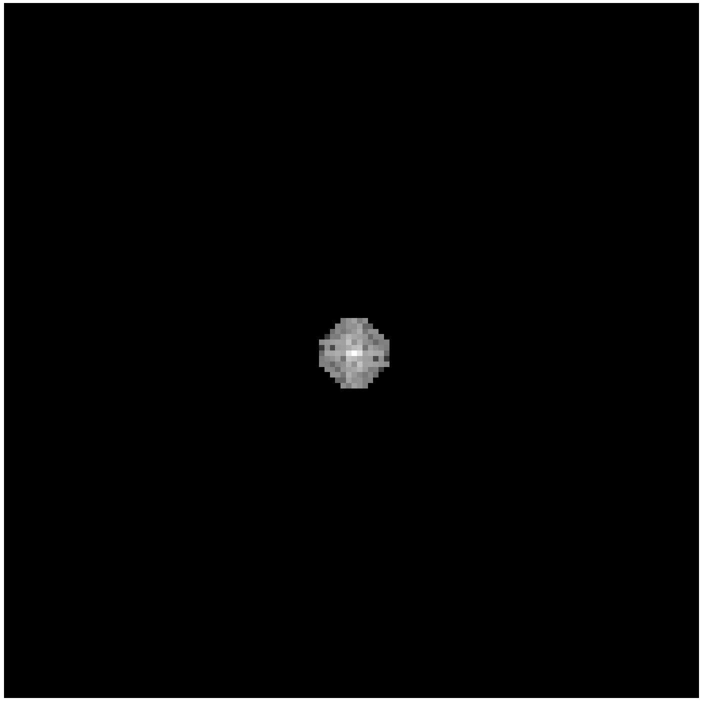


An image (500x500 pixels) and its Fourier spectrum. The super-imposed circles have radii values of 5, 15, 30, 80, and 230, which respectively enclose 92.0, 94.6, 96.4, 98.0, and 99.5% of the image power.



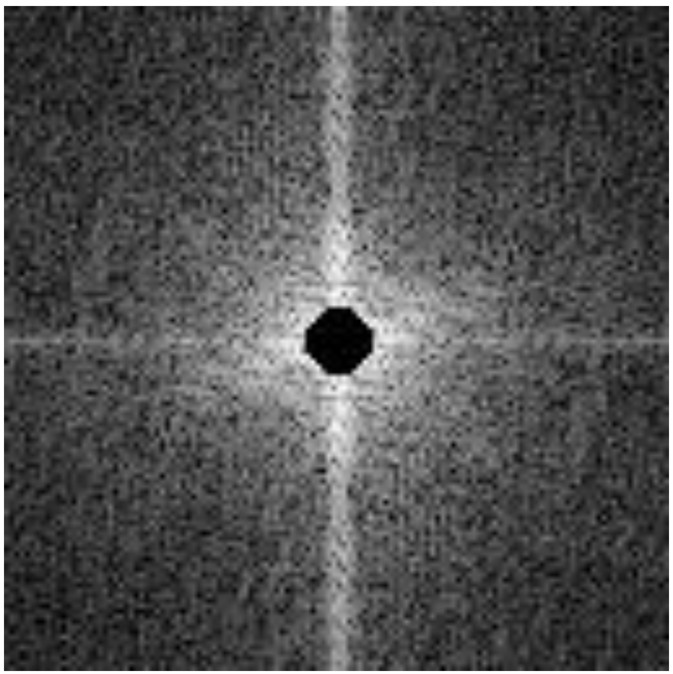
If I discard high-frequencies, I get a blurred image...
Why?

Low-Pass Filtered



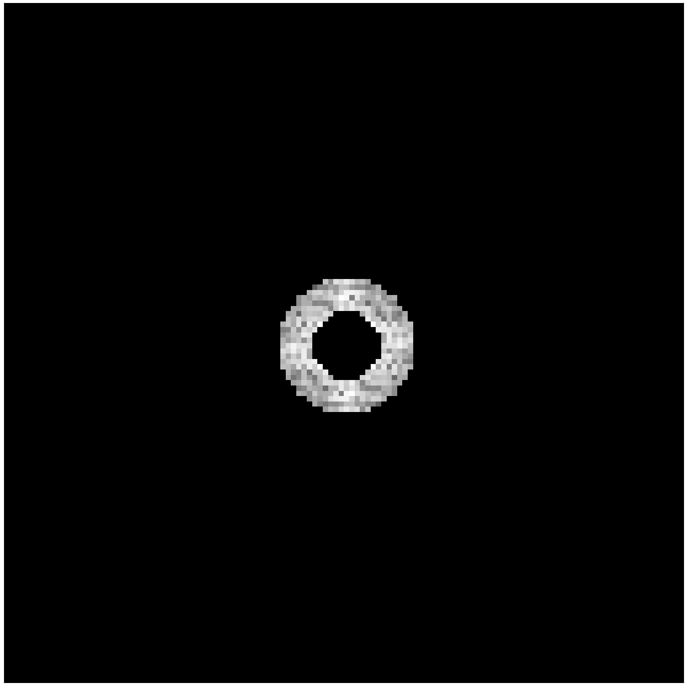
Inverse Transformed

High-Pass Filtered



Inverse Transformed

Band-Pass Filtered



Inverse Transformed

Why bother with FT?

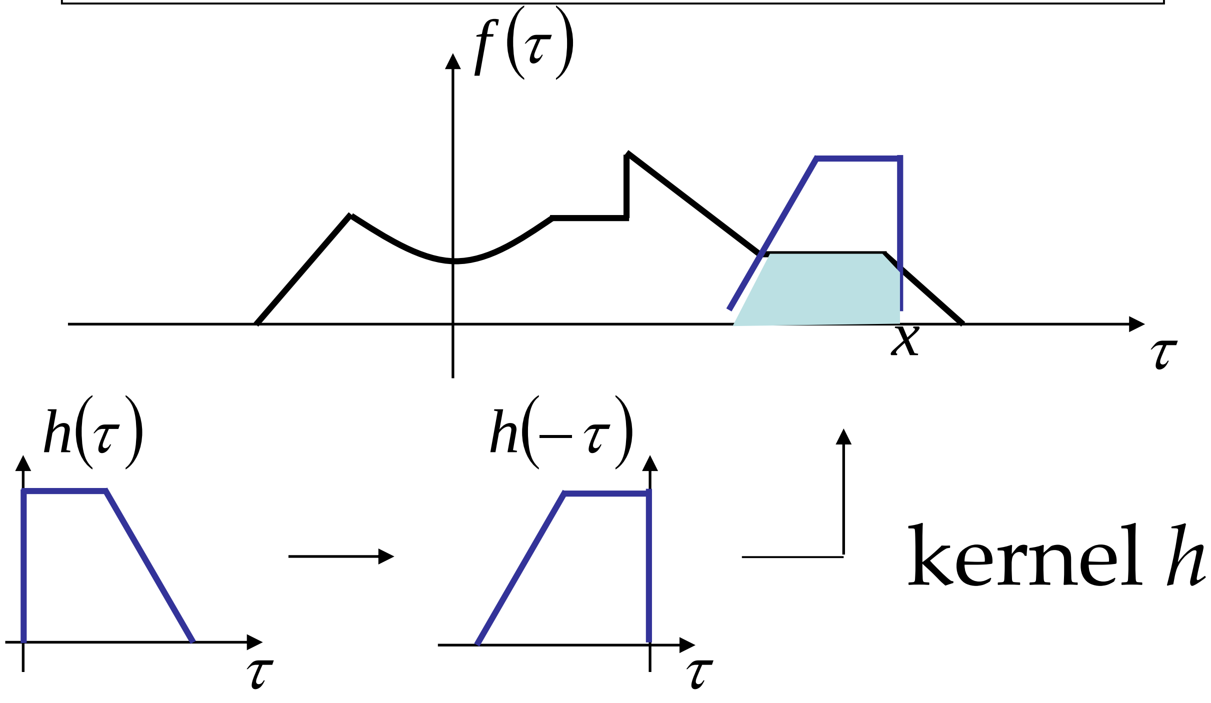
- Great for filtering.
- Great for compression.
- In some situations: Much faster than operating in the spatial domain.
- Convolutions are simple multiplications in Frequency space!
- ...

Outline

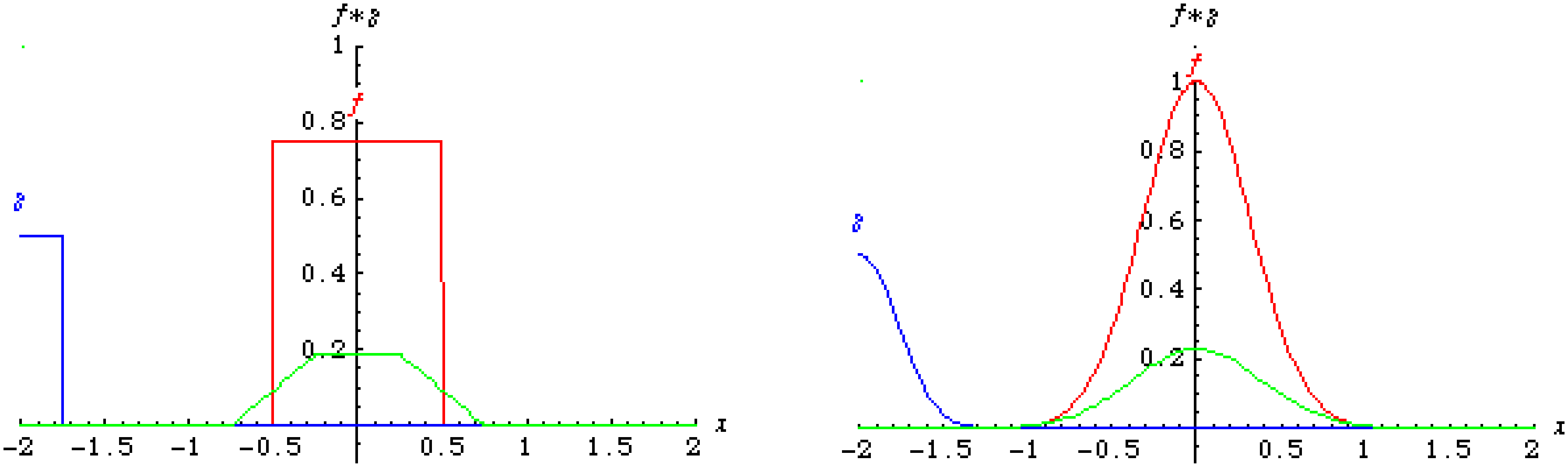
- Single Pixel Manipulation
- Frequency Space
  - Fourier Transform
  - Frequency Space
  - Spatial Convolution
- Digital Filters

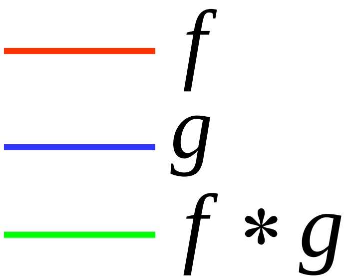
Convolution

$$g(x) = \int_{-\infty}^{\infty} f(\tau) h(x-\tau) d\tau \qquad g = f * h$$



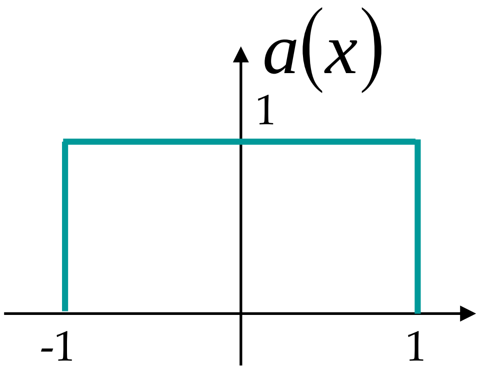
Convolution - Example

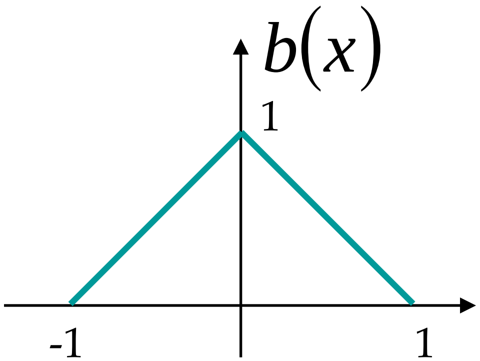




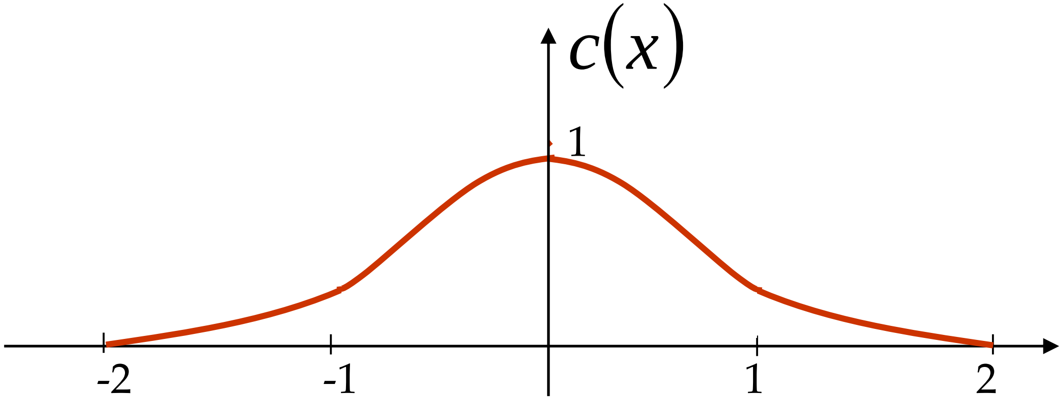
Eric Weinstein's Math World

Convolution - Example





$$c = a * b$$



Properties of Convolution

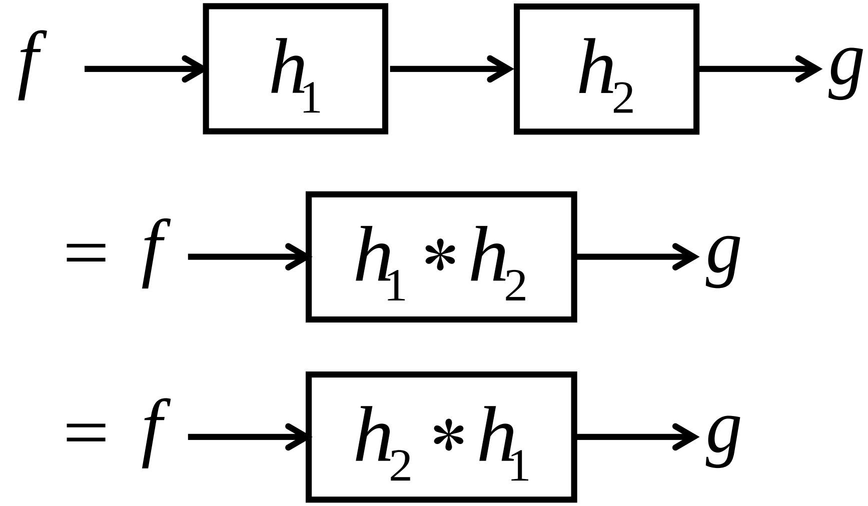
- Commutative

$$a * b = b * a$$

- Associative

$$(a * b) * c = a * (b * c)$$

- Cascade system



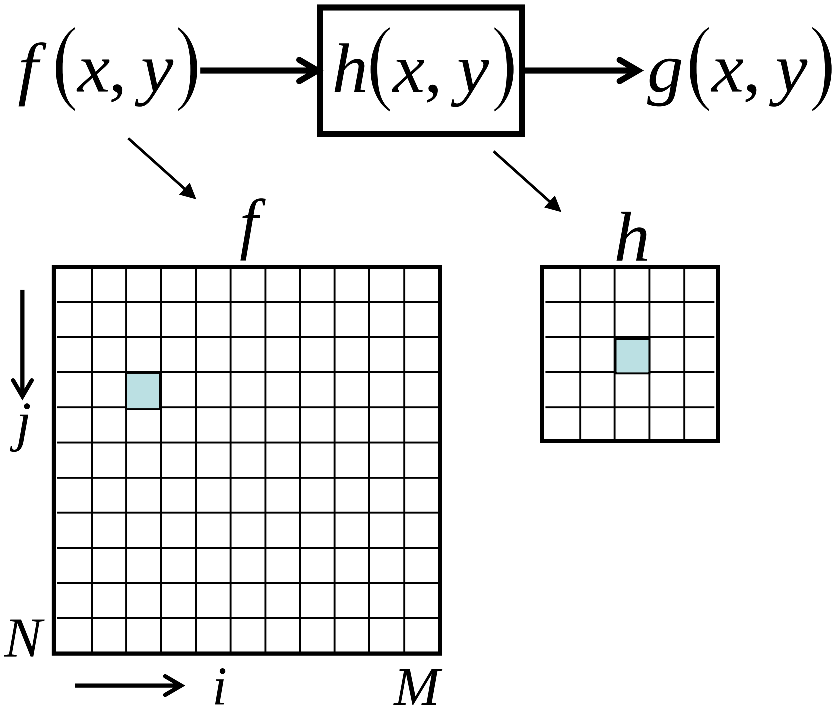
Outline

- Single Pixel Manipulation
- Frequency Space
- Digital Filters
  - Spatial filters
  - Frequency domain filtering
  - Edge detection

Outline

- Single Pixel Manipulation
- Frequency Space
- Digital Filters
  - Spatial filters
  - Frequency domain filtering
  - Edge detection

Images are Discrete and Finite



Convolution

$$g(i,j)=\sum_{m=1}^{M}\sum_{n=1}^{N} f(m,n)\,h(i-m,j-n)$$

Fourier Transform

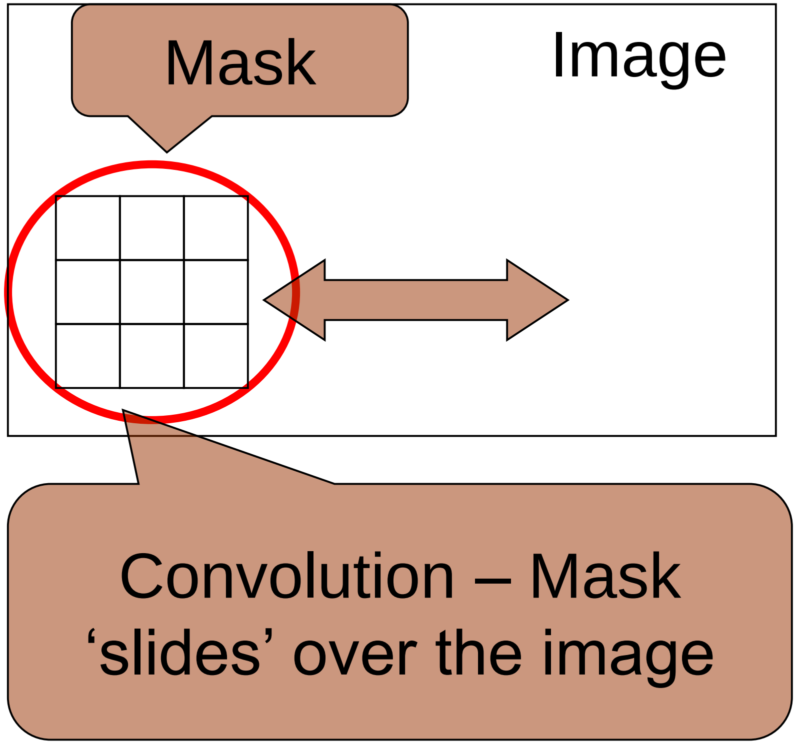
$$F(u,v)=\sum_{m=1}^{M}\sum_{n=1}^{N} f(m,n)\,e^{-i2\pi\left(\frac{mu}{M}+\frac{nv}{N}\right)}$$

Inverse Fourier Transform

$$f(k,l)=\frac{1}{MN}\sum_{u=1}^{M}\sum_{v=1}^{N} F(u,v)\,e^{i2\pi\left(\frac{ku}{M}+\frac{lv}{N}\right)}$$

Spatial Mask

- Simple way to process an image.
- Mask defines the processing function.
- Corresponds to a multiplication in frequency domain.



Example

- Each mask position has weight **w**.
- The result of the operation for each pixel is given by:

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

Mask

2	2	2
4	4	4
4	5	6

Image

$$g(x,y) = \sum_{s=-a}^{a} \sum_{t=-b}^{b} w(s,t) f(x+s, y+t)$$

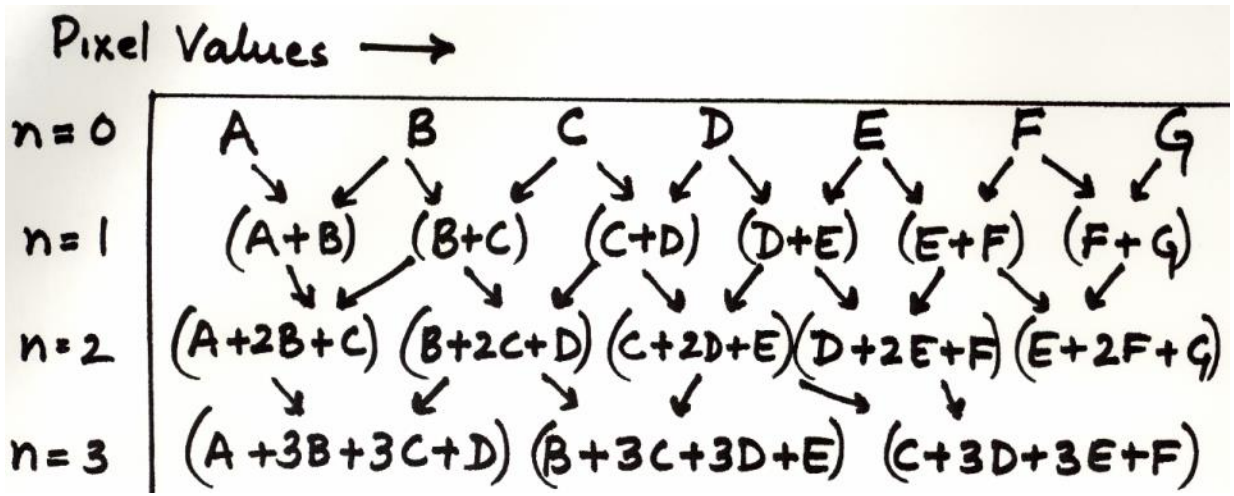
=1*2+2*2+1*2+…
=8+0-20
=-12

Definitions

- Spatial filters
  - Use a **mask (kernel)** over an image region.
  - Work directly with pixels.
  - As opposed to: **Frequency filters**.
- Advantages
  - Simple implementation: **convolution** with the kernel function.
  - Different masks offer a **large variety of functionalities**.

Averaging

Let's think about averaging pixel values



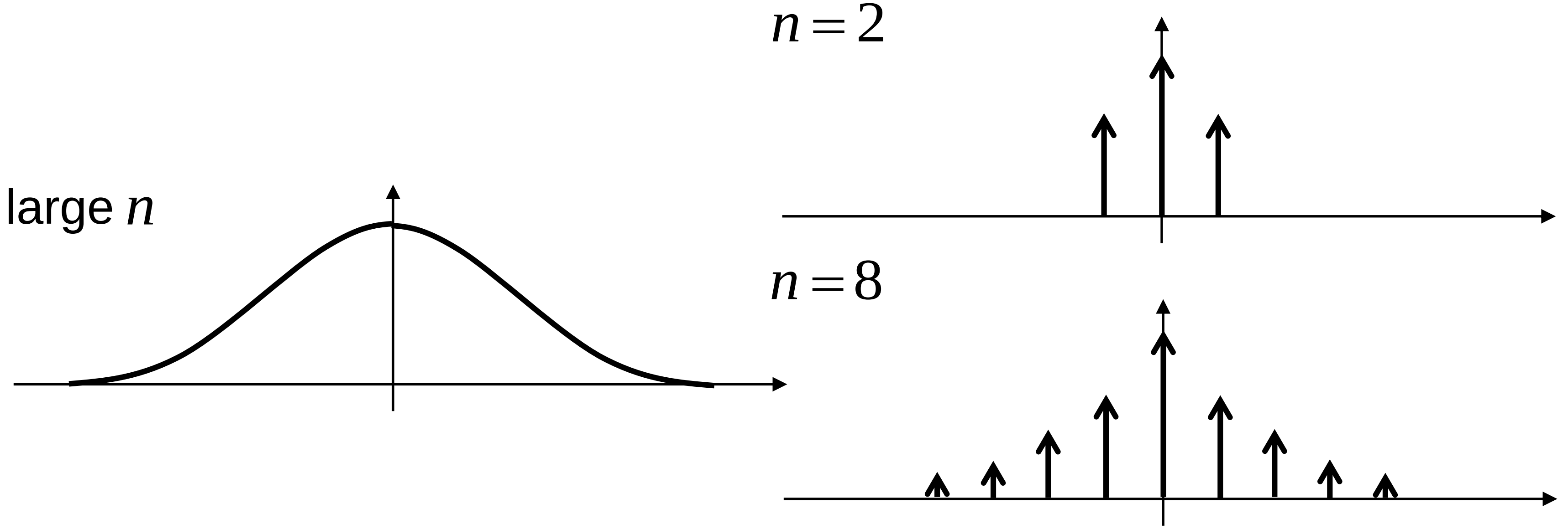
For *n=2*, convolve pixel values with $\begin{bmatrix} 1 & 2 & 1 \end{bmatrix}$

2D images:

(a) use $\begin{bmatrix} 1 & 2 & 1 \end{bmatrix}$ then $\begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix}$ or (b) use $\begin{bmatrix} 1 & 2 & 1 \end{bmatrix} * \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$

Averaging

The convolution kernel



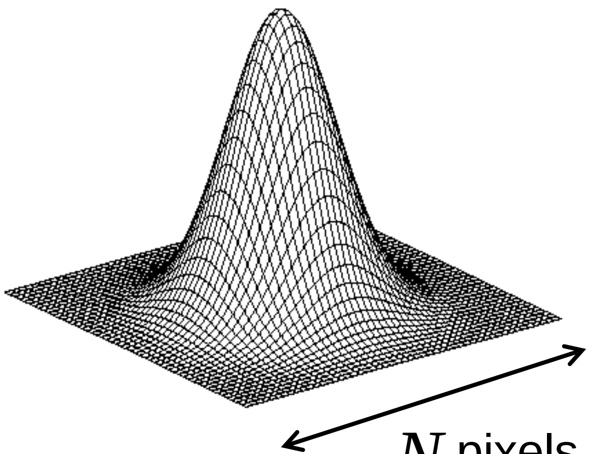
Repeated averaging ≈ Gaussian smoothing

Gaussian Smoothing

Gaussian kernel

$$h(i,j)=\frac{1}{2\pi\sigma^2}e^{-\frac{1}{2}\left(\frac{i^2+j^2}{\sigma^2}\right)}$$

Filter size $N \propto \sigma$ …can be very large (truncate, if necessary)



$$g(i,j)=\frac{1}{2\pi\sigma^2}\sum_{m=1}\sum_{n=1}e^{-\frac{1}{2}\left(\frac{m^2+n^2}{\sigma^2}\right)}f(i-m,j-n)$$

2D Gaussian is separable!

$$g(i,j)=\frac{1}{2\pi\sigma^2}\sum_{m=1}e^{-\frac{1}{2}\frac{m^2}{\sigma^2}}\sum_{n=1}e^{-\frac{1}{2}\frac{n^2}{\sigma^2}}f(i-m,j-n)$$

Use two 1D Gaussian Filters!

Gaussian Smoothing

- A Gaussian kernel gives less weight to pixels further from the center of the window

$H[u,v]$

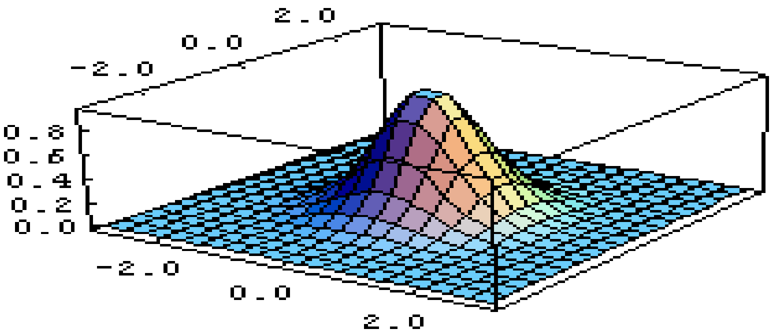
$$\frac{1}{16}$$

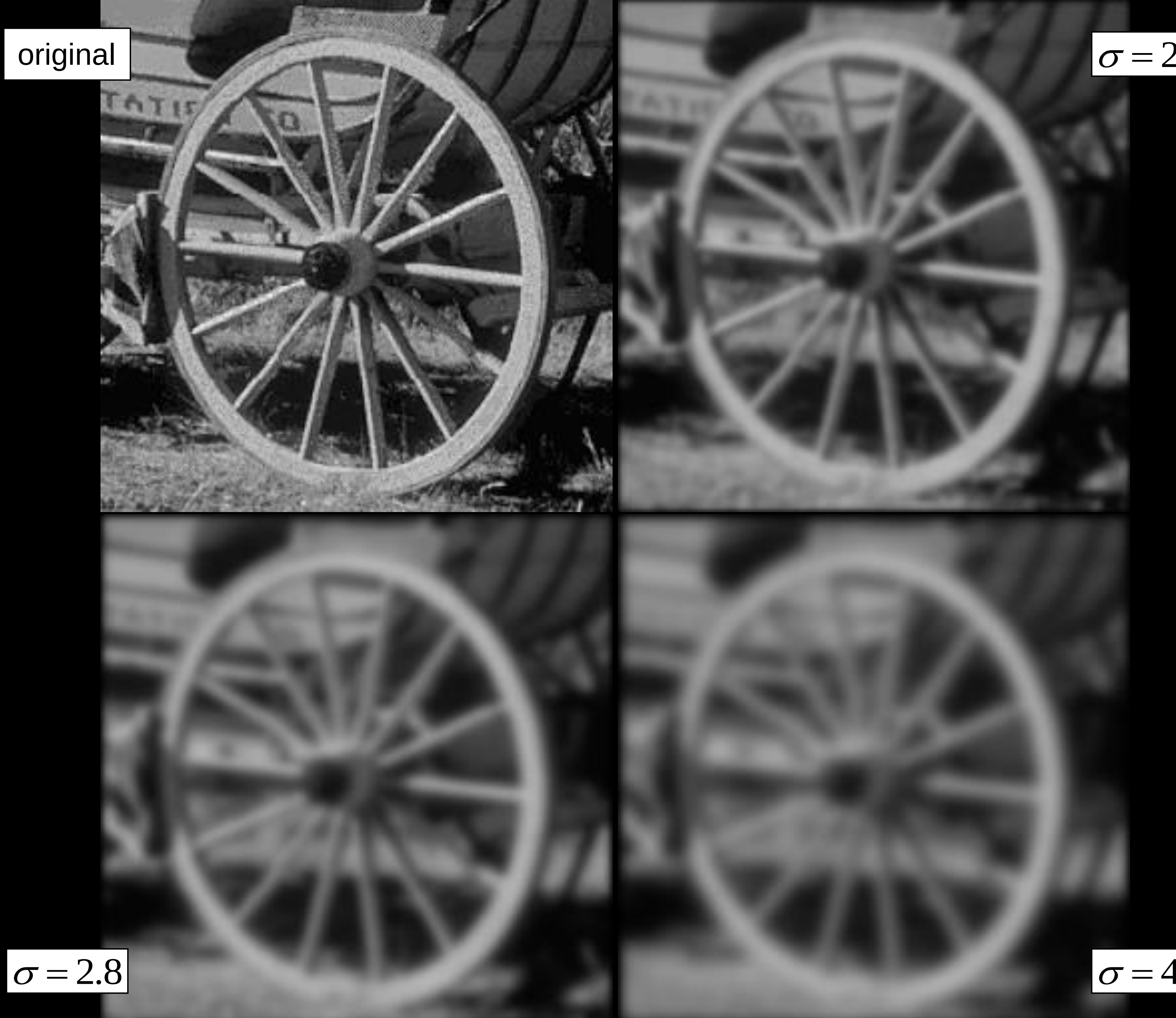
1	2	1
2	4	2
1	2	1

- This kernel is an approximation of a Gaussian function:

$F[x,y]$

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

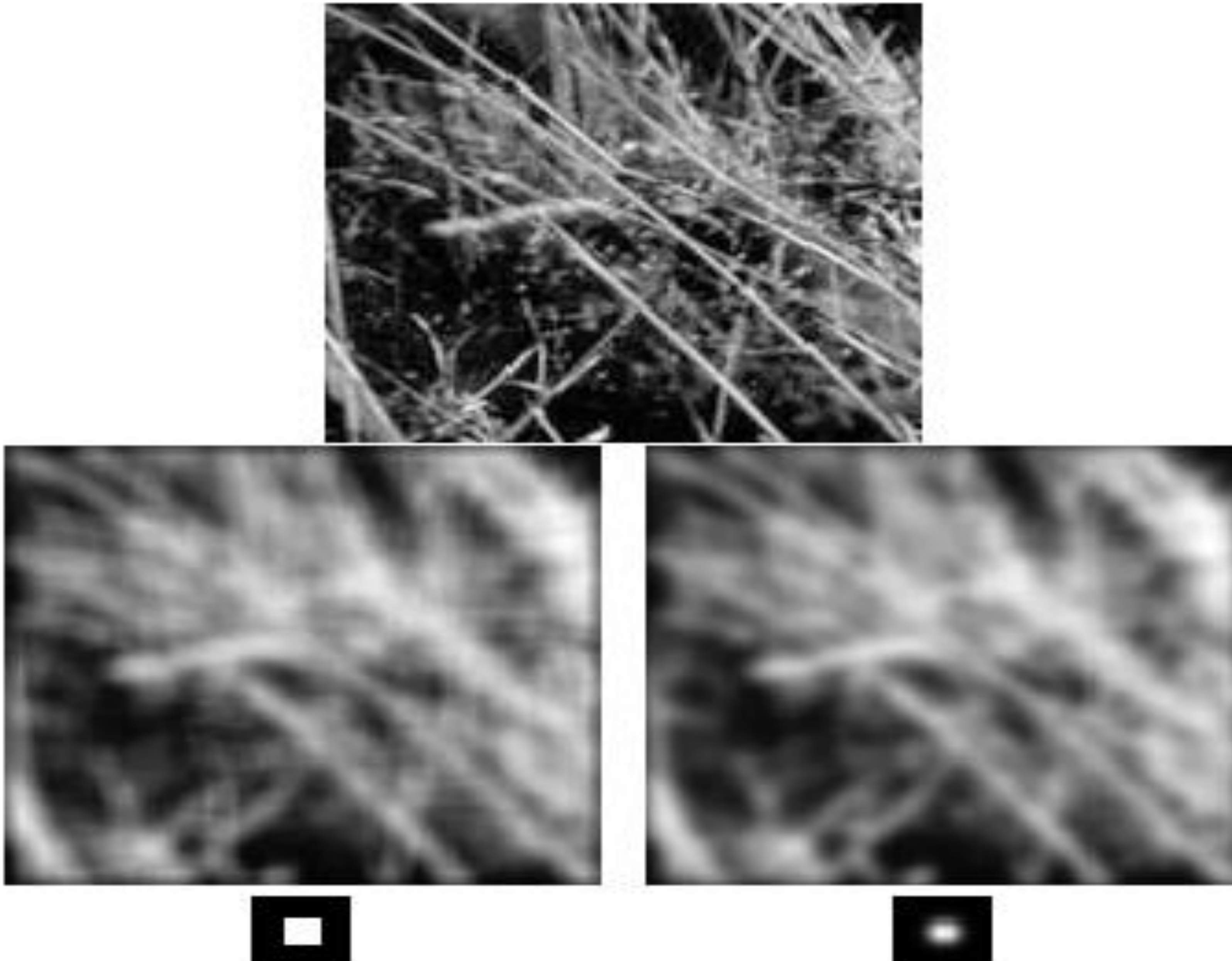



original
$\sigma = 2$
$\sigma = 2.8$
$\sigma = 4$

Mean Filtering

- We are degrading the energy of the high spatial frequencies of an image (**low-pass filtering**).
  - Makes the image ‘smoother’.
  - Used in noise reduction.
- Can be implemented with spatial masks or in the frequency domain.

1/9	1/9	1/9
1/9	1/9	1/9
1/9	1/9	1/9



Mean filter

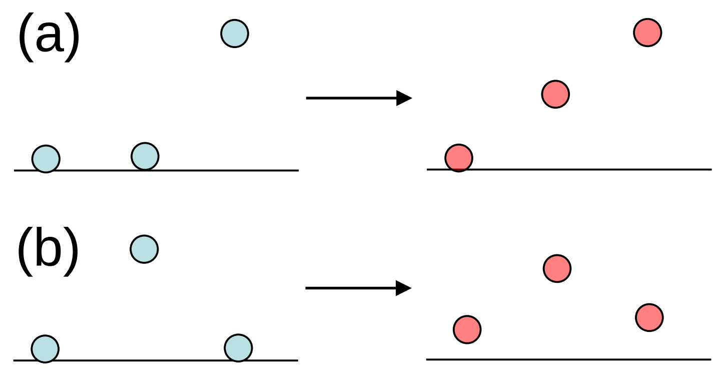
Gaussian filter

http://www.michaelbach.de/ot/cog_blureffects/index.html

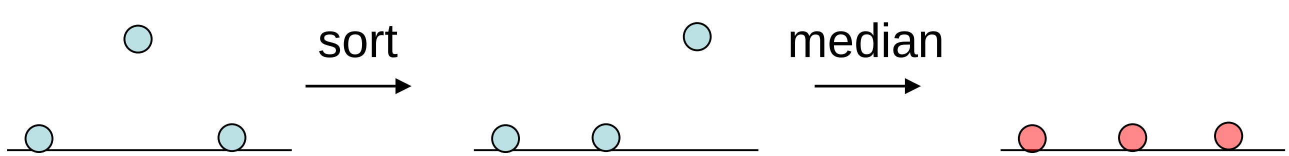
http://www.michaelbach.de/ot/cog_blureffects/index.html

Median Filter

- Smoothing is averaging
  - (a) Blurs edges
  - (b) Sensitive to outliers



- Median filtering
  - – Sort $N^2 - 1$ values around the pixel
  - – Select middle value (median)



  - – Non-linear (Cannot be implemented with convolution)

Median Filter

x = 3

3	9	4	52	3	8	6	2	2	9

9

y[1] = median[3 3 9] = 3
y[2] = median[3 4 9] = 4
y[3] = median[4 9 52] = 9
y[4] = median[3 4 52] = 4
y[5] = median[3 8 52] = 8

y[6] = median[3 6 8] = 6
y[7] = median[2 6 8] = 6
y[8] = median[2 2 6] = 2
y[9] = median[2 2 9] = 2
y[10] = median[2 9 9] = 9

y =

3	4	9	4	8	6	6	2	2	9

Median Filter

Sorted: 0,0,1,1,1,2,2,4,4

Input

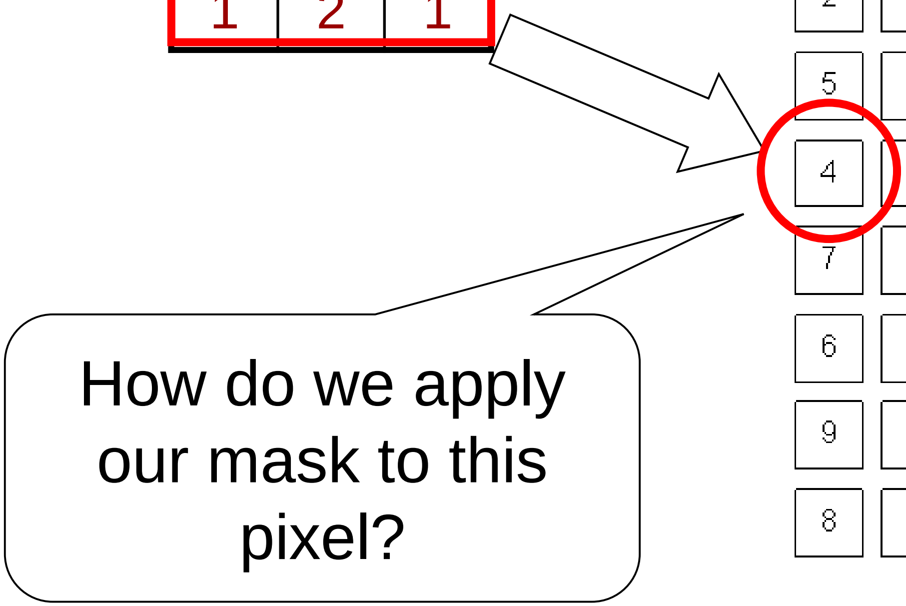
1	4	0	1	3	1
2	2	4	2	2	3
1	0	1	0	1	0
1	2	1	0	2	2
2	5	3	1	2	5
1	1	4	2	3	0

Output

1	4	0	1	3	1
2	1	1	1	1	3
1	1	1	1	2	0
1	1	1	1	1	2
2	2	2	2	2	5
1	1	4	2	3	0

Salt and pepper noise **Gaussian noise**

Border Problem



$\frac{1}{16}$

1	2	1
2	4	2
1	2	1

0	3	2	5	4	7	6	9	8
3	0	1	2	3	4	5	6	7
2	1	0	3	2	5	4	7	6
5	2	3	0	1	2	3	4	5
4	3	2	1	0	3	2	5	4
7	4	5	2	3	0	1	2	3
6	5	4	3	2	1	0	3	2
9	6	7	4	5	2	3	0	1
8	7	6	5	4	3	2	1	0

What a computer sees

Border Problem

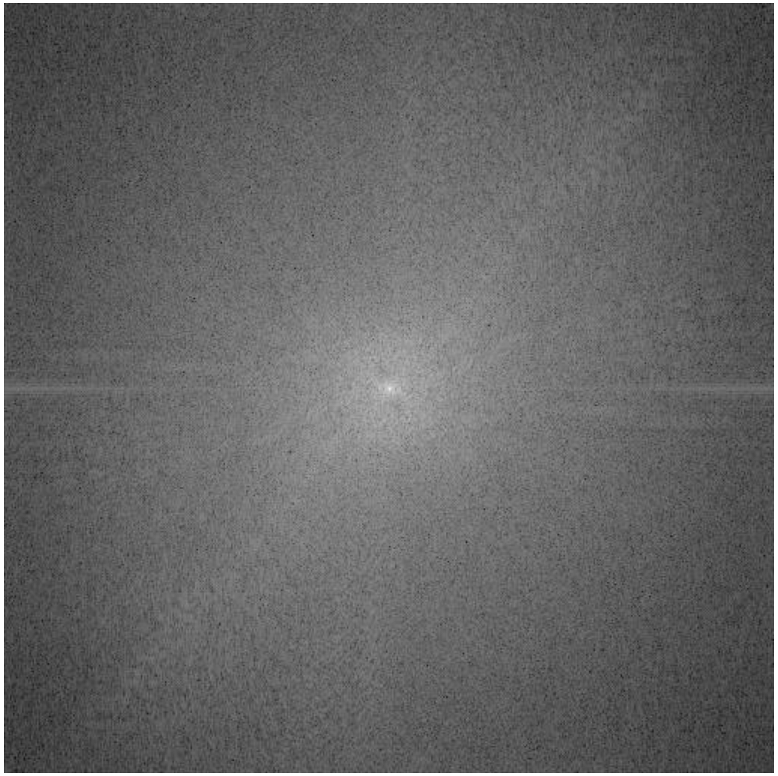
- Ignore
  - Output image will be smaller than original
- Pad with constant values
  - Can introduce substantial 1^{st} order derivative values
- Pad with reflection
  - Can introduce substantial 2^{nd} order derivative values

Outline

- Single Pixel Manipulation
- Frequency Space
- Digital Filters
  - Spatial filters
  - Frequency domain filtering
  - Edge detection

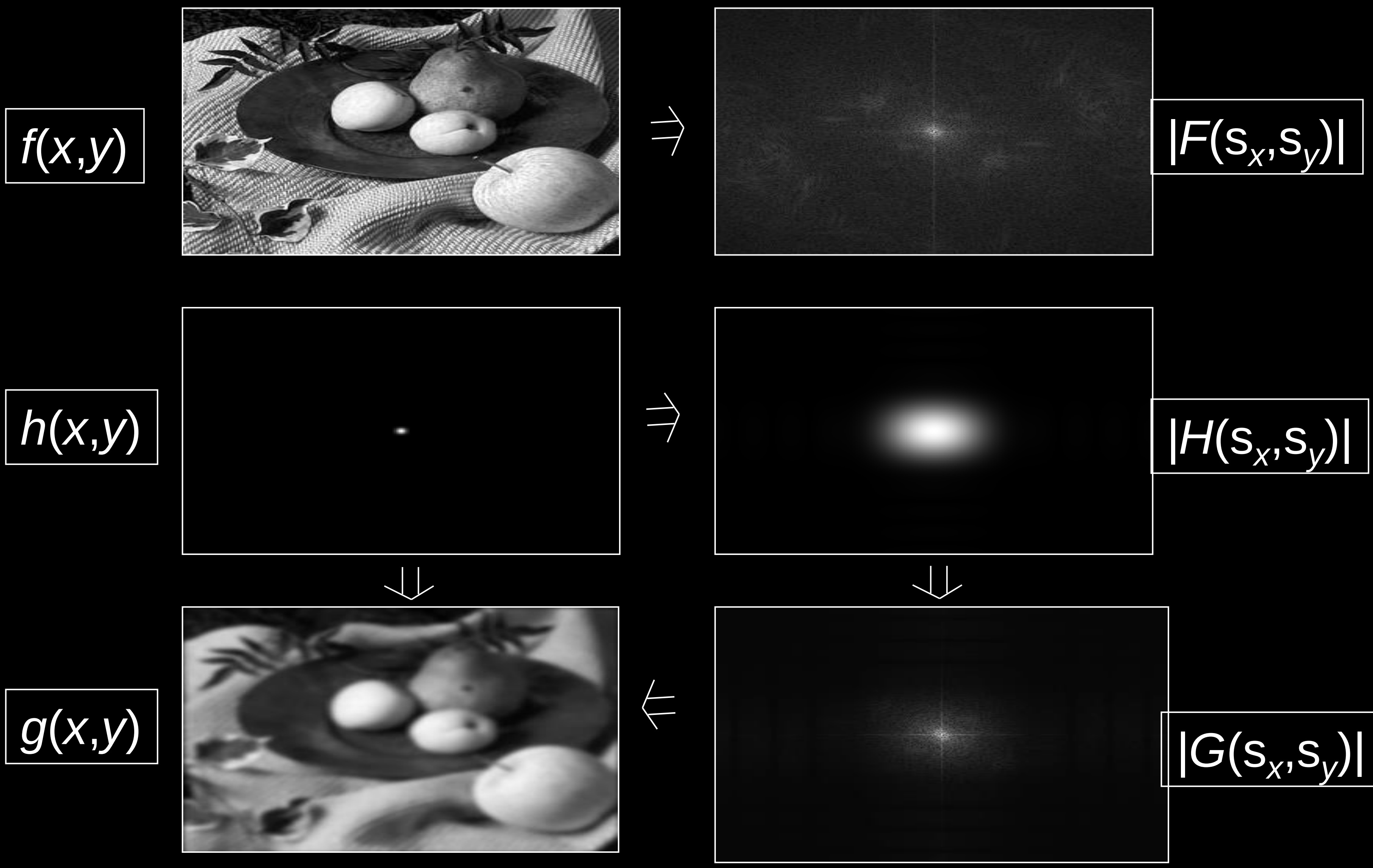
Image Processing in the Fourier Domain

Magnitude of the FT

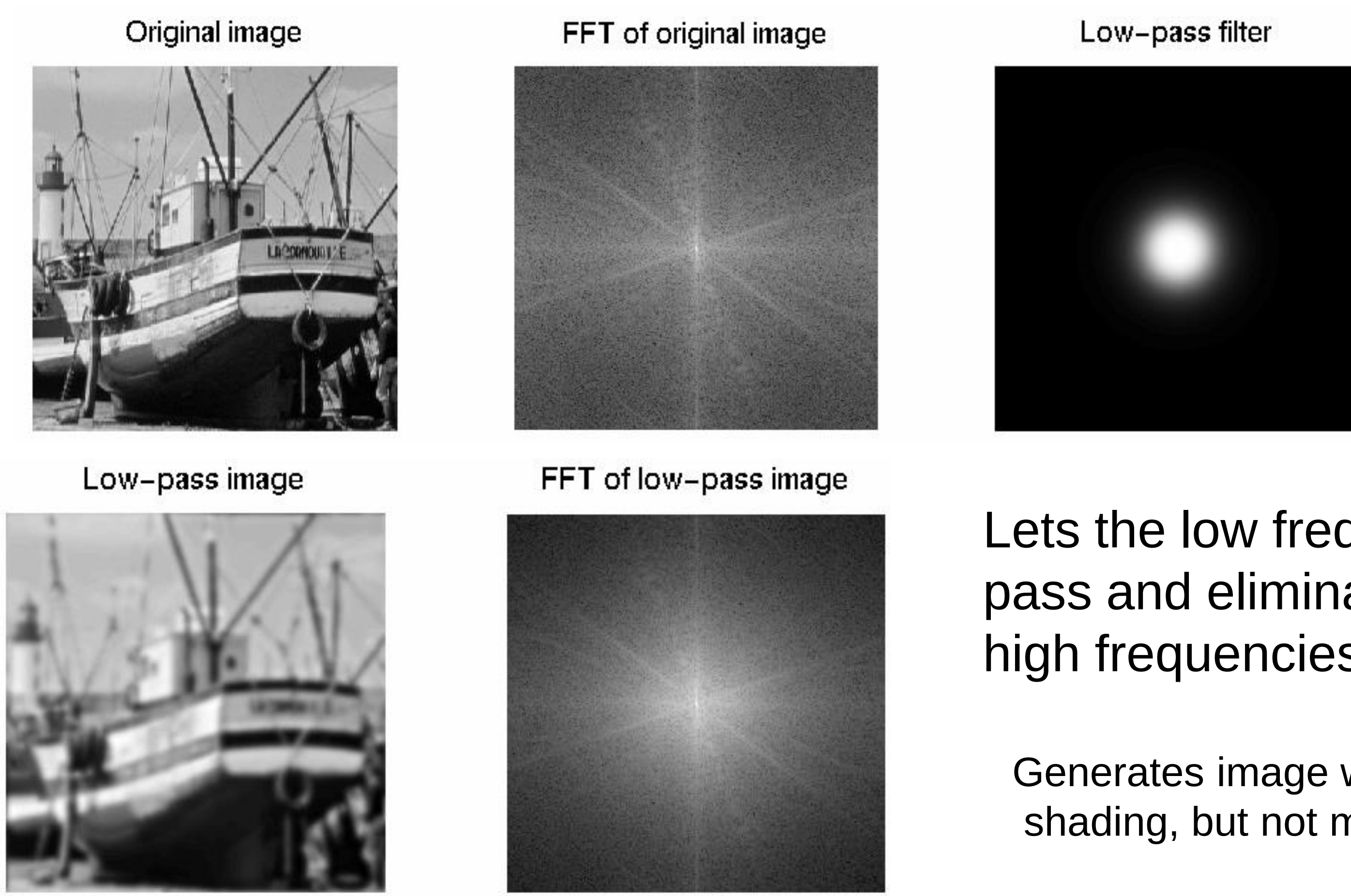


Does not look anything like what we have seen

Convolution in the Frequency Domain



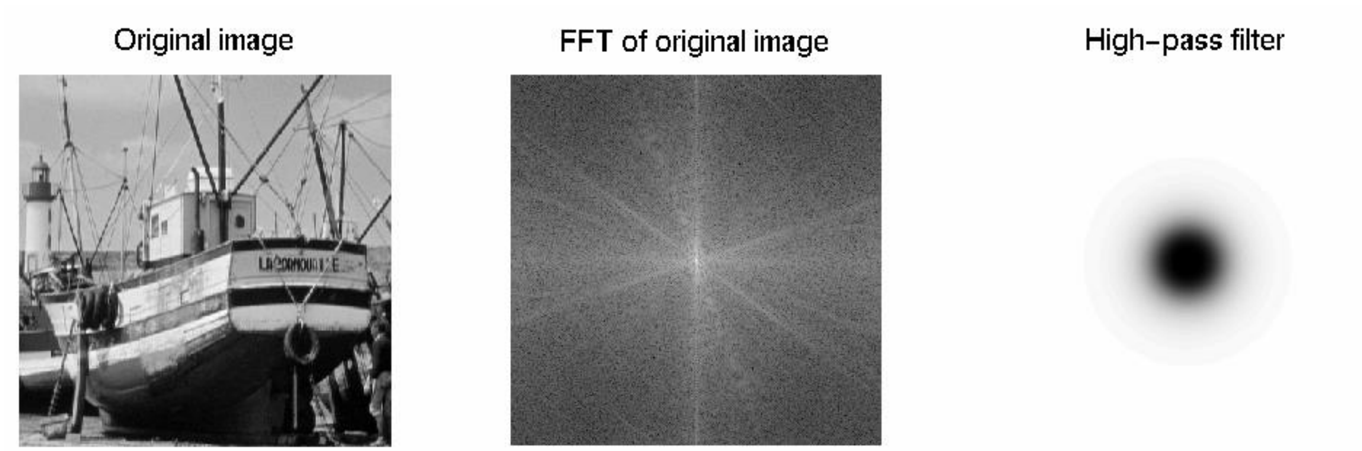
Low-pass Filtering

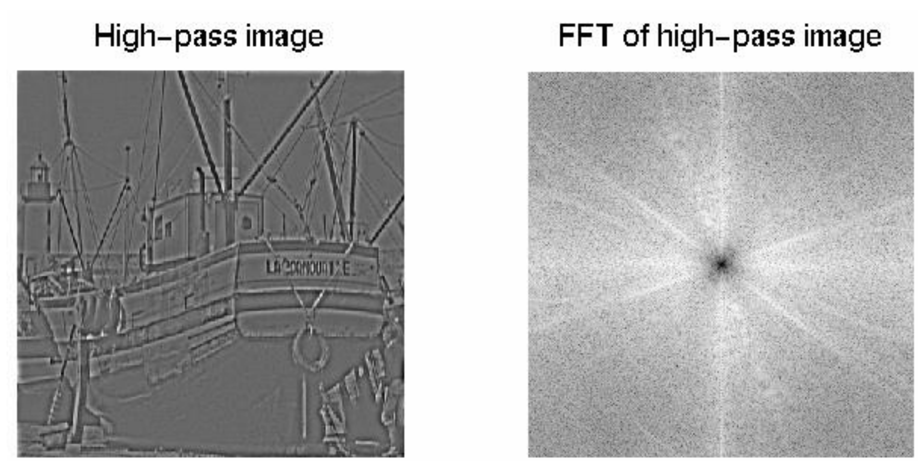


Lets the low frequencies pass and eliminates the high frequencies.

Generates image with overall shading, but not much detail

High-pass Filtering





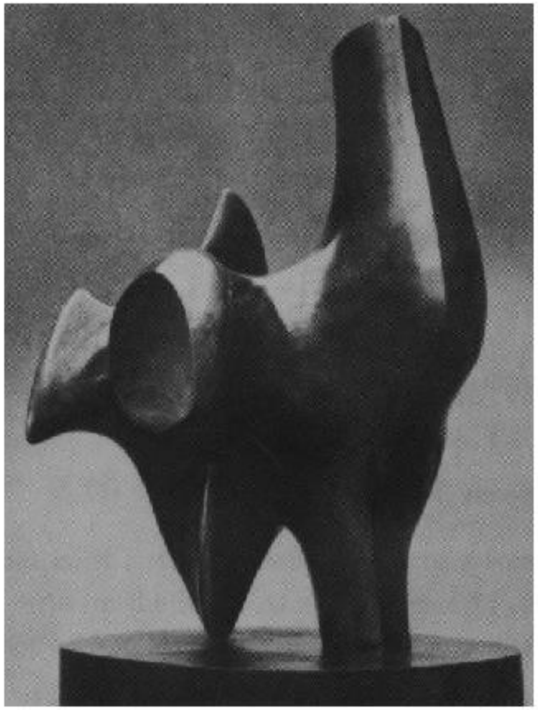
Lets through the high frequencies (the detail), but eliminates the low frequencies (the overall shape). It acts like an edge enhancer.

Outline

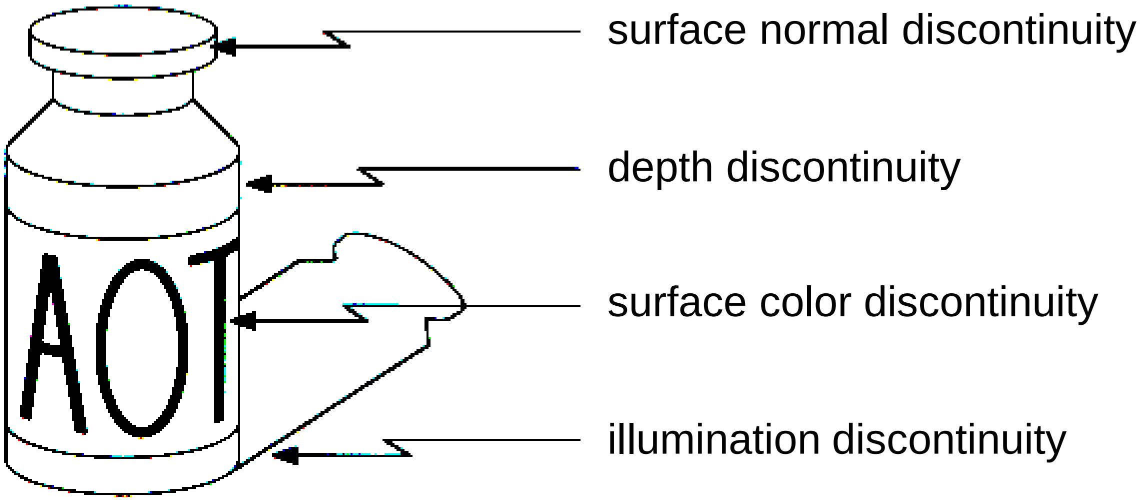
- Single Pixel Manipulation
- Frequency Space
- Digital Filters
  - Spatial filters
  - Frequency domain filtering
  - Edge detection

Edge Detection

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

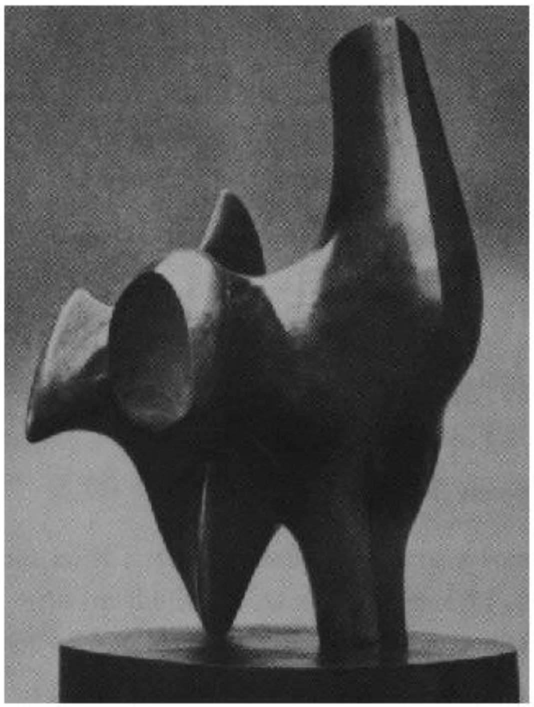


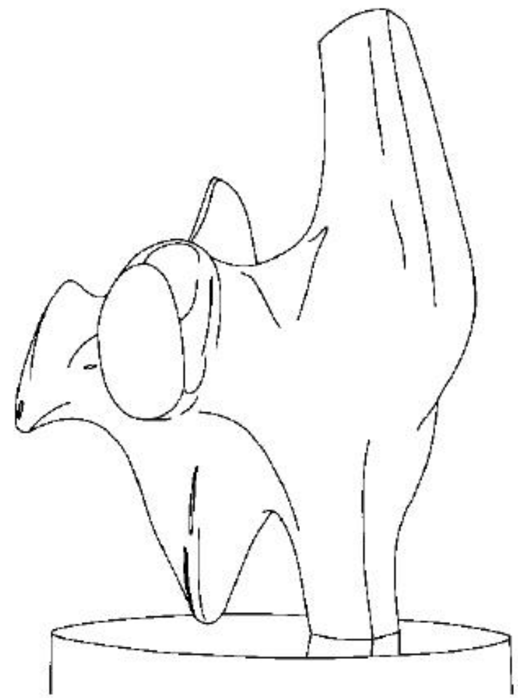
Origin of Edges



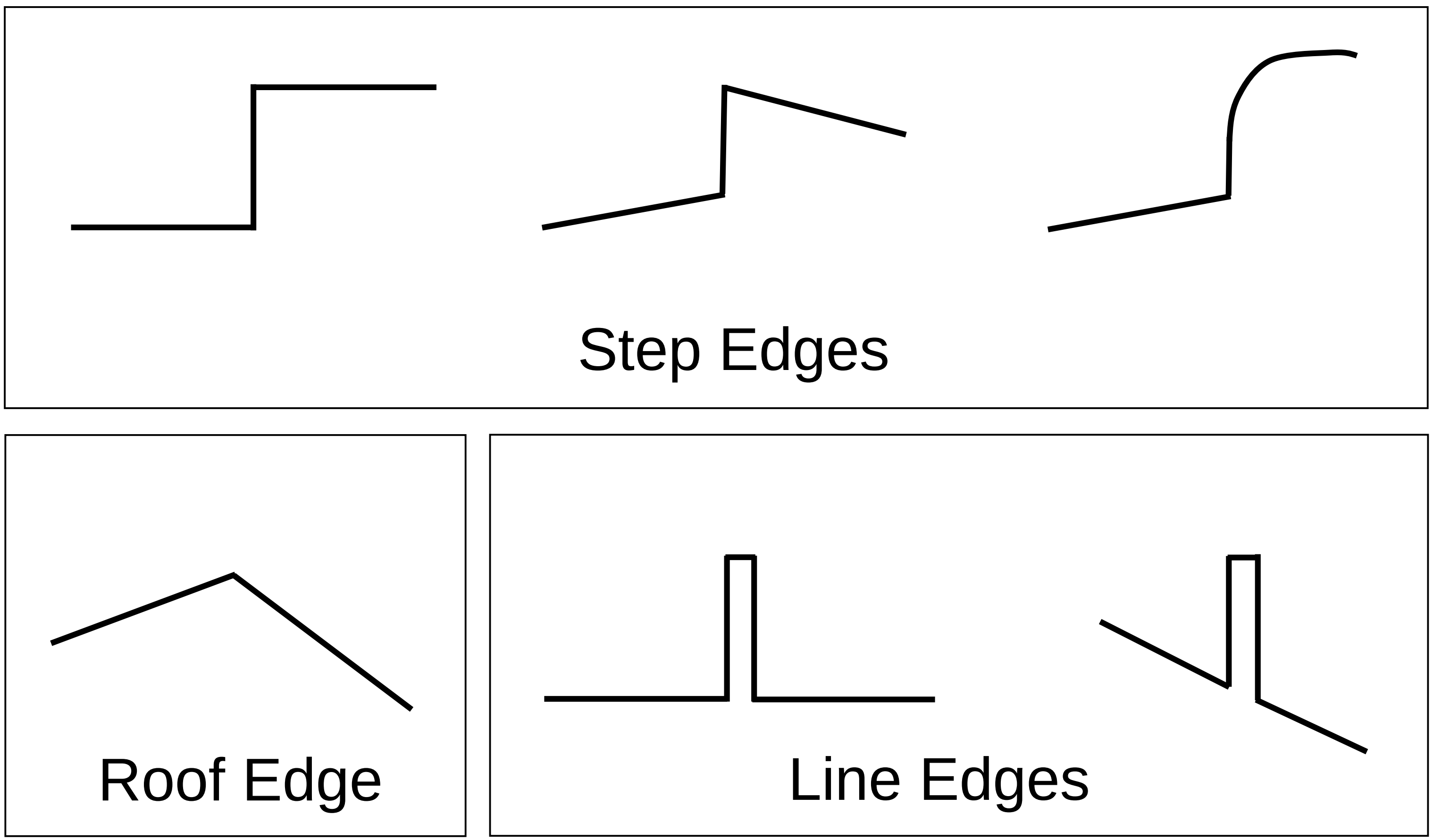
- Edges are caused by a variety of factors

How can you tell that a pixel is on an edge?

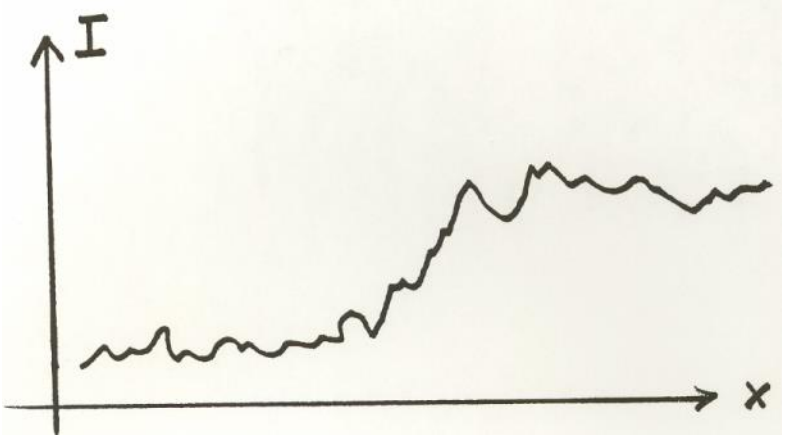




Edge Types



Real Edges



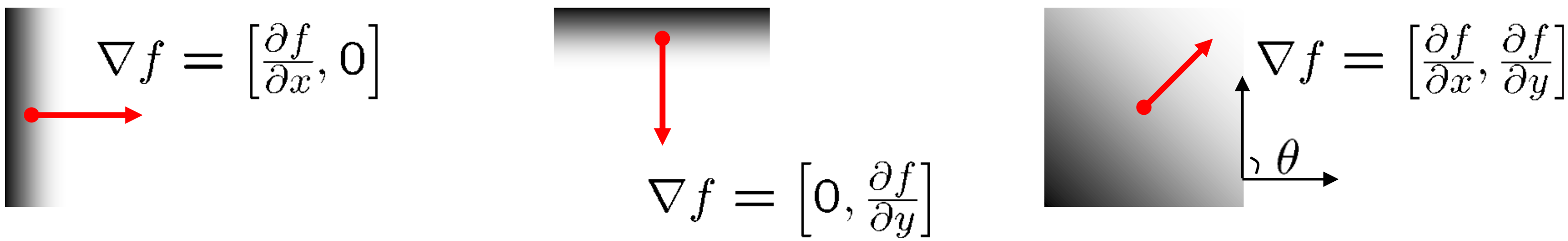
Noisy and Discrete!

We want an **Edge Operator** that produces:

- Edge Magnitude
- Edge Orientation
- High Detection Rate and Good Localization

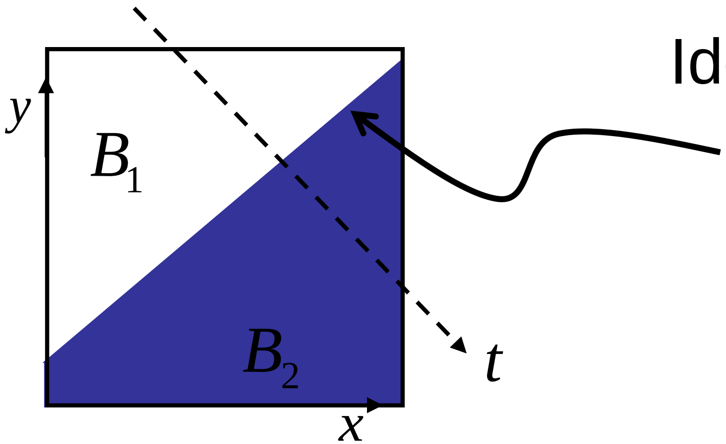
Gradient

- Gradient equation: $\nabla f = \left[\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right]$
- Represents direction of most rapid change in intensity



- Gradient direction: $\theta = \tan^{-1}\left(\frac{\partial f}{\partial y} / \frac{\partial f}{\partial x}\right)$
- The *edge strength* is given by the gradient magnitude $\|\nabla f\| = \sqrt{\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2}$

Theory of Edge Detection



Ideal edge

$$L(x,y) = x\sin\theta - y\cos\theta + \rho = 0$$

$$B_1 : L(x,y) < 0$$

$$B_2 : L(x,y) > 0$$

Unit step function:

$$u(t) = \begin{cases} 1 & \text{for } t > 0 \\ 1/2 & \text{for } t = 0 \\ 0 & \text{for } t < 0 \end{cases} \qquad u(t) = \int_{-\infty}^{t} \delta(s)\,ds$$

Image intensity (brightness):

$$I(x,y) = B_1 + (B_2 - B_1)\,u(x\sin\theta - y\cos\theta + \rho)$$

Theory of Edge Detection

- Partial derivatives (gradients):

$$\frac{\partial I}{\partial x} = +\sin\theta\left(B_2 - B_1\right)\delta\left(x\sin\theta - y\cos\theta + \rho\right)$$

$$\frac{\partial I}{\partial y} = -\cos\theta\left(B_2 - B_1\right)\delta\left(x\sin\theta - y\cos\theta + \rho\right)$$

- Squared gradient:

$$s(x,y) = \left(\frac{\partial I}{\partial x}\right)^2 + \left(\frac{\partial I}{\partial y}\right)^2 = \left[\left(B_2 - B_1\right)\delta\left(x\sin\theta - y\cos\theta + \rho\right)\right]^2$$

Edge Magnitude: $\sqrt{s(x,y)}$

Edge Orientation: $\arctan\left(\frac{\partial I}{\partial y} / \frac{\partial I}{\partial x}\right)$ (normal of the edge)

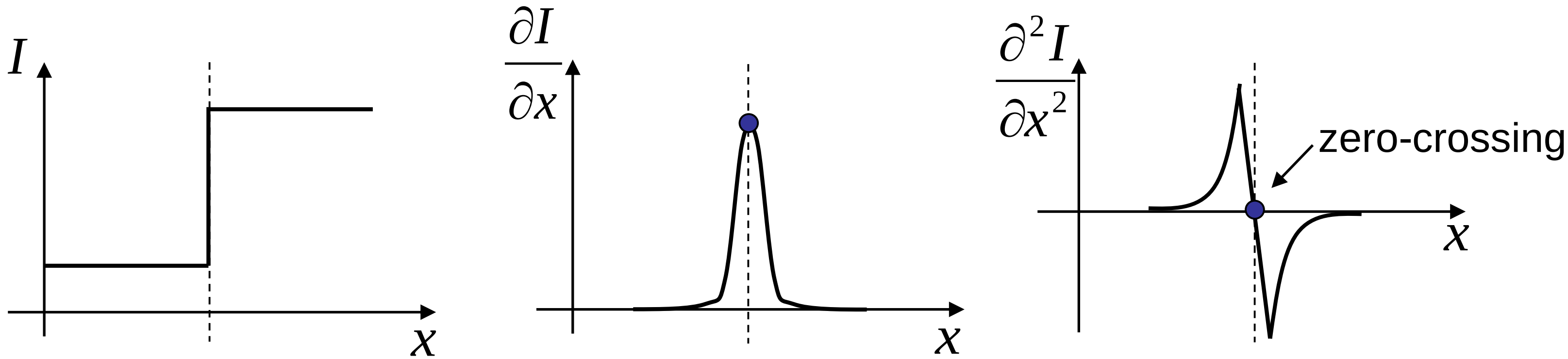
Rotationally symmetric, non-linear operator

Theory of Edge Detection

- Laplacian:

$$\nabla^2 I = \frac{\partial^2 I}{\partial x^2} + \frac{\partial^2 I}{\partial y^2} = (B_2 - B_1)\delta'(x \sin\theta - y\cos\theta + \rho)$$

Rotationally symmetric, linear operator



Discrete Edge Operators

- How can we differentiate a ***discrete*** image?

Finite difference approximations:

$$\frac{\partial I}{\partial x} \approx \frac{1}{2\varepsilon}\left(\left(I_{i+1,j+1} - I_{i,j+1}\right) + \left(I_{i+1,j} - I_{i,j}\right)\right)$$

$$\frac{\partial I}{\partial y} \approx \frac{1}{2\varepsilon}\left(\left(I_{i+1,j+1} - I_{i+1,j}\right) + \left(I_{i,j+1} - I_{i,j}\right)\right)$$

$I_{i,j+1}$	$I_{i+1,j+1}$
$I_{i,j}$	$I_{i+1,j}$

ε

Convolution masks :

$$\frac{\partial I}{\partial x} \approx \frac{1}{2\varepsilon} \begin{array}{|c|c|}\hline -1 & 1 \\ \hline -1 & 1 \\ \hline \end{array} \qquad \frac{\partial I}{\partial y} \approx \frac{1}{2\varepsilon} \begin{array}{|c|c|}\hline 1 & 1 \\ \hline -1 & -1 \\ \hline \end{array}$$

Discrete Edge Operators

- Second order partial derivatives:

$$\frac{\partial^2 I}{\partial x^2} \approx \frac{1}{\varepsilon^2}\left(I_{i-1,j} - 2I_{i,j} + I_{i+1,j}\right)$$

$$\frac{\partial^2 I}{\partial y^2} \approx \frac{1}{\varepsilon^2}\left(I_{i,j-1} - 2I_{i,j} + I_{i,j+1}\right)$$

$I_{i-1,j+1}$	$I_{i,j+1}$	$I_{i+1,j+1}$
$I_{i-1,j}$	$I_{i,j}$	$I_{i+1,j}$
$I_{i-1,j-1}$	$I_{i,j-1}$	$I_{i+1,j-1}$

- **Laplacian** :

$$\nabla^2 I = \frac{\partial^2 I}{\partial x^2} + \frac{\partial^2 I}{\partial y^2}$$

Convolution masks :

$$\nabla^2 I \approx \frac{1}{\varepsilon^2}$$

0	1	0
1	−4	1
0	1	0

or

$$\frac{1}{6\varepsilon^2}$$

1	4	1
4	−20	4
1	4	1

(more accurate)

The Sobel Operators

- Better approximations of the gradients exist
  - The *Sobel* operators below are commonly used

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

s_x

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

s_y

Comparing Edge Operators

Gradient:

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

Good Localization
Noise Sensitive
Poor Detection

Roberts (2 x 2):

0	1
-1	0

1	0
0	-1

Sobel (3 x 3):

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

1	2	1
0	0	0
-1	-2	1

Sobel (5 x 5):

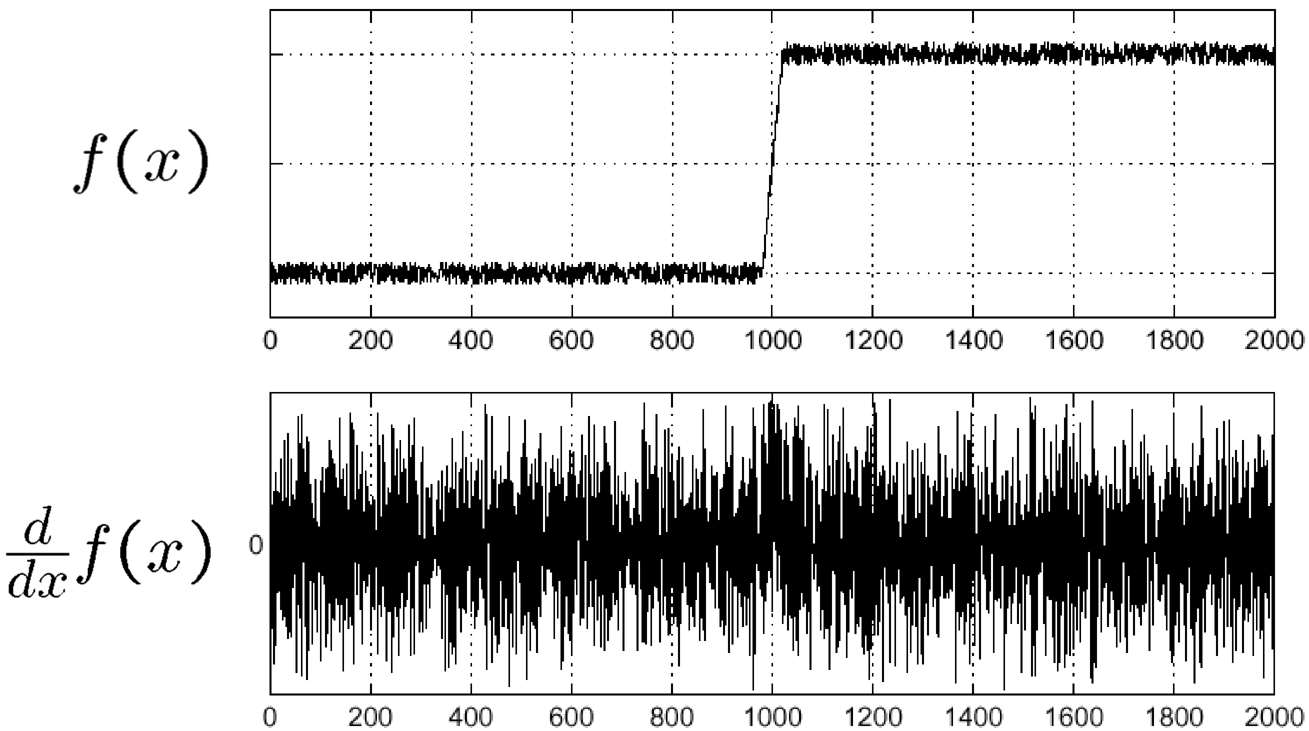
-1	-2	0	2	1
-2	-3	0	3	2
-3	-5	0	5	3
-2	-3	0	3	2
-1	-2	0	2	1

1	2	3	2	1
2	3	5	3	2
0	0	0	0	0
-2	-3	-5	-3	-2
-1	-2	-3	-2	-1

Poor Localization
Less Noise Sensitive
Good Detection

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??

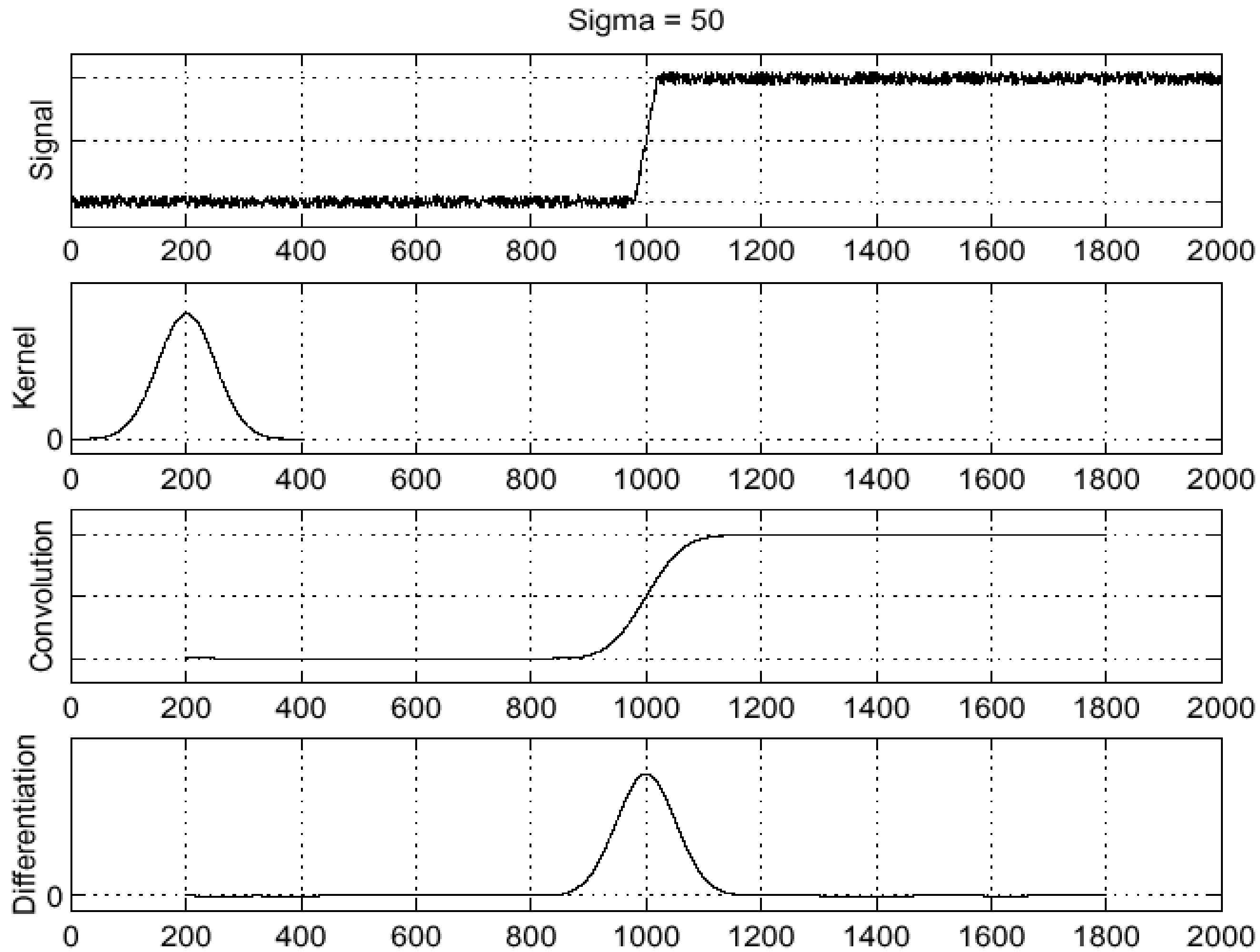
Solution: Smooth First

f

h

$h \star f$

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



Where is the edge?

Look for peaks in $\frac{\partial}{\partial x}(h \star f)$

Derivative Theorem of Convolution

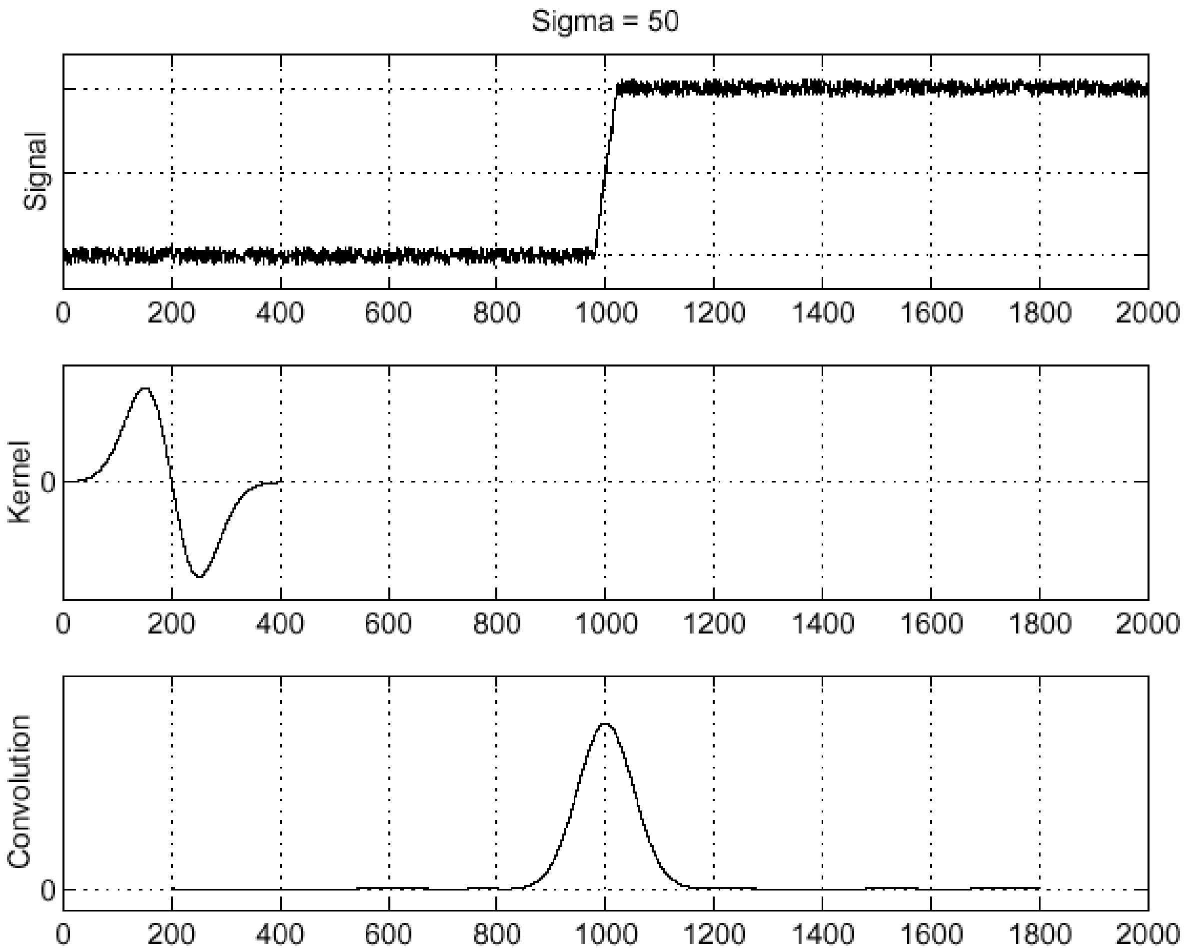
$$\frac{\partial}{\partial x}(h \star f) = (\frac{\partial}{\partial x}h) \star f$$

…saves us one operation.

f

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

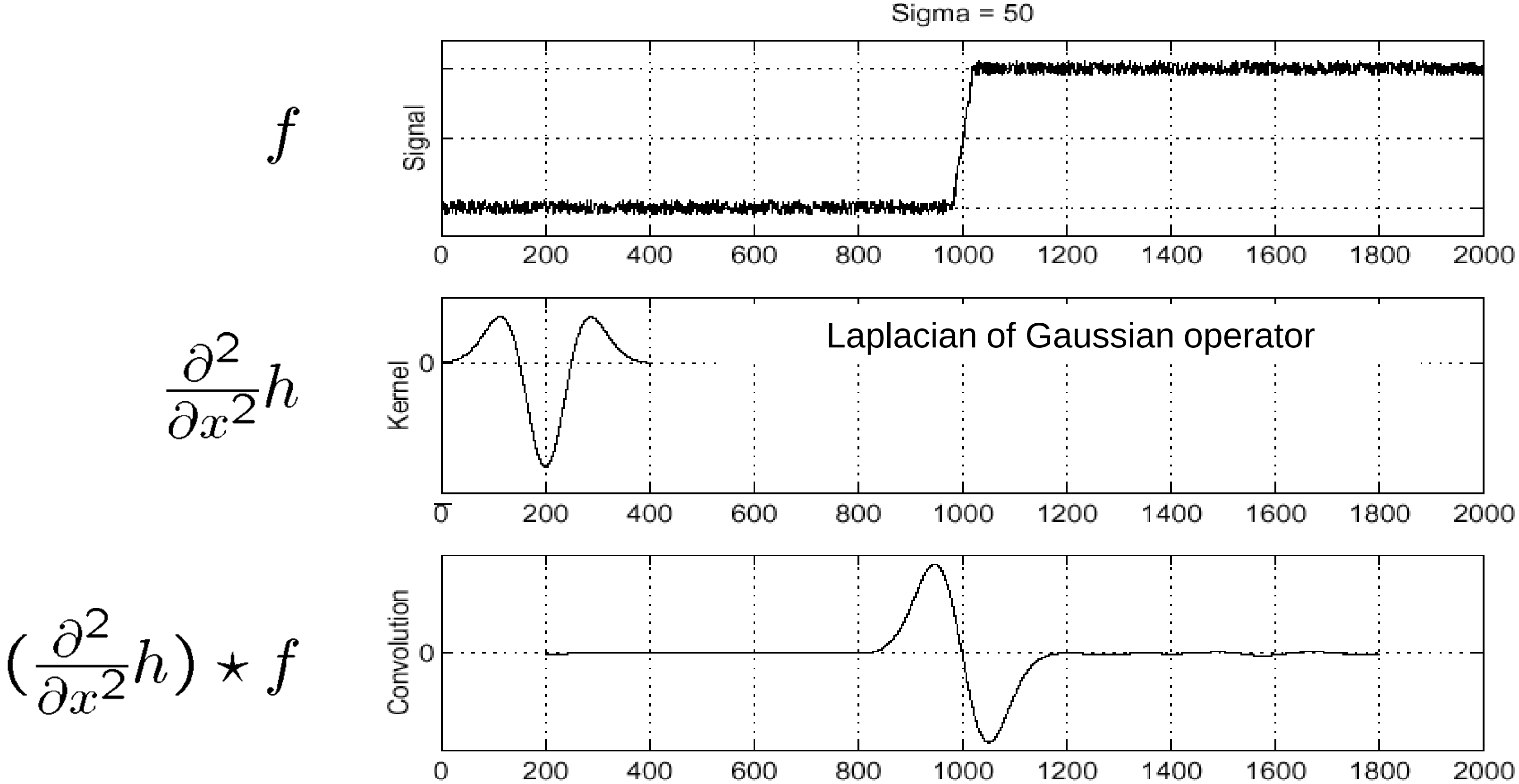
$(\frac{\partial}{\partial x}h) \star f$



Laplacian of Gaussian (LoG)

$$\frac{\partial^2}{\partial x^2}(h * f) = \left(\frac{\partial^2}{\partial x^2} h\right) * f$$

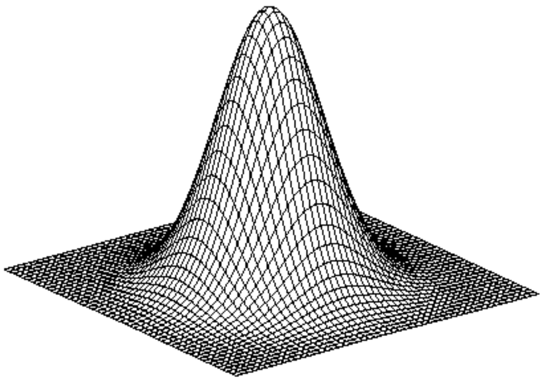
Laplacian of Gaussian



Where is the edge?

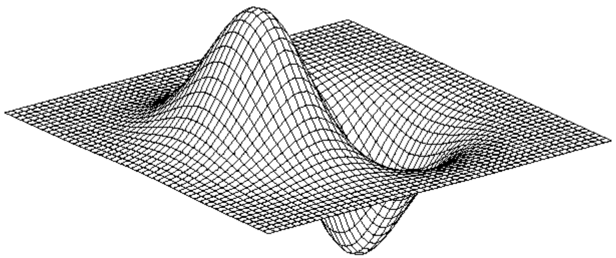
Zero-crossings of bottom graph !

2D Gaussian Edge Operators



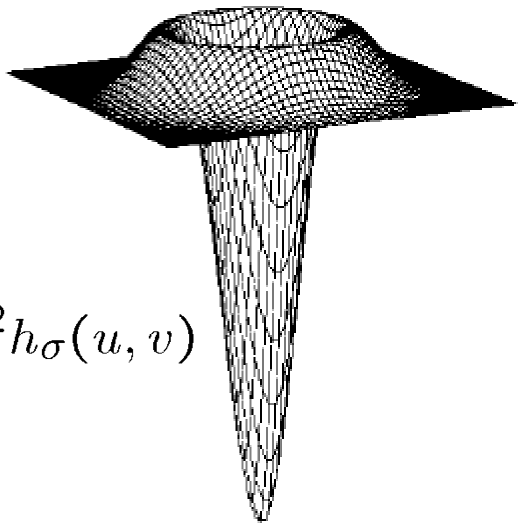
$$h_\sigma(u,v) = \frac{1}{2\pi\sigma^2} e^{-\frac{u^2+v^2}{2\sigma^2}}$$

Gaussian



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

Derivative of Gaussian (DoG)



$$\nabla^2 h_\sigma(u,v)$$

Laplacian of Gaussian

Mexican Hat (Sombrero)

- ∇^2 is the **Laplacian** operator: $\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$

Canny Edge Operator

- Smooth image *I* with 2D Gaussian: $G * I$

- Find local edge normal directions for each pixel

$$\overline{\mathbf{n}} = \frac{\nabla (G * I)}{\left|\nabla (G * I)\right|}$$

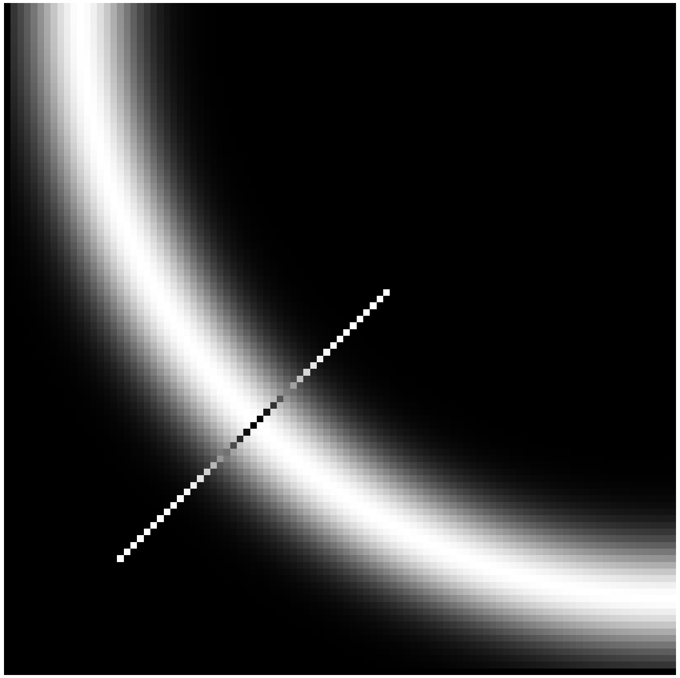
- Compute edge magnitudes $\left|\nabla (G * I)\right|$

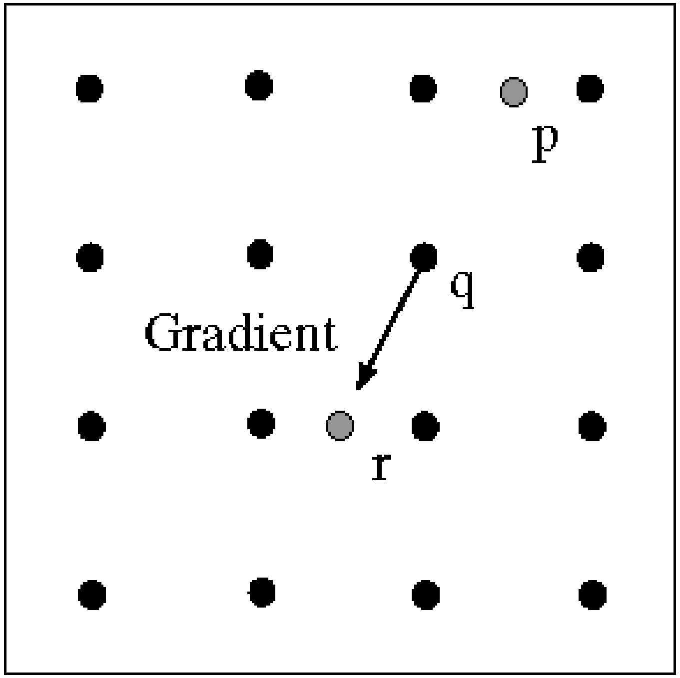
- Locate edges by finding zero-crossings along the edge normal directions (**non-maximum suppression**)

$$\frac{\partial^2 (G * I)}{\partial \overline{\mathbf{n}}^2} = 0$$

Non-maximum Suppression

- Check if pixel is local maximum along gradient direction
  - requires checking interpolated pixels p and r





original image

magnitude of the gradient

After non-maximum suppression

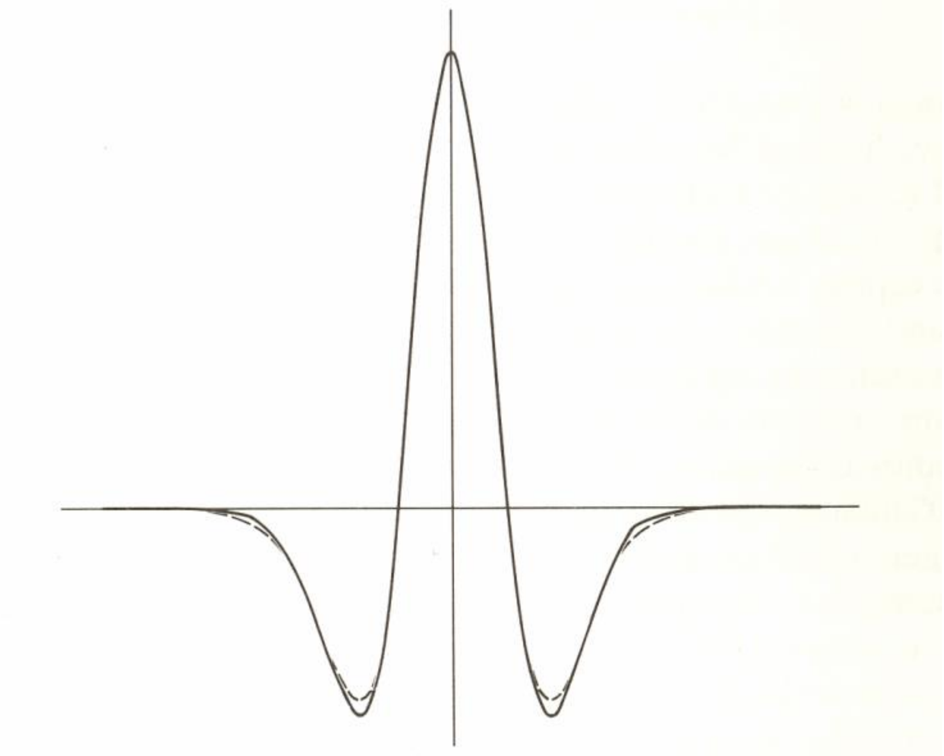
Canny Edge Operator

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 features

Difference of Gaussians (DoG)

- Laplacian of Gaussian can be approximated by the difference between two different Gaussians



DoG Edge Detection

(a) $\sigma = 1$ (b) $\sigma = 2$ (b)-(a)

Outline

- Single Pixel Manipulation
- Frequency Space
- Digital Filters

Acknowledgements: Most of this course is based on the excellent courses offered by Prof. Shree Nayar at Columbia University, USA and by Prof. Srinivasa Narasimhan at CMU, USA. Please acknowledge the original source when reusing these slides for academic purposes.