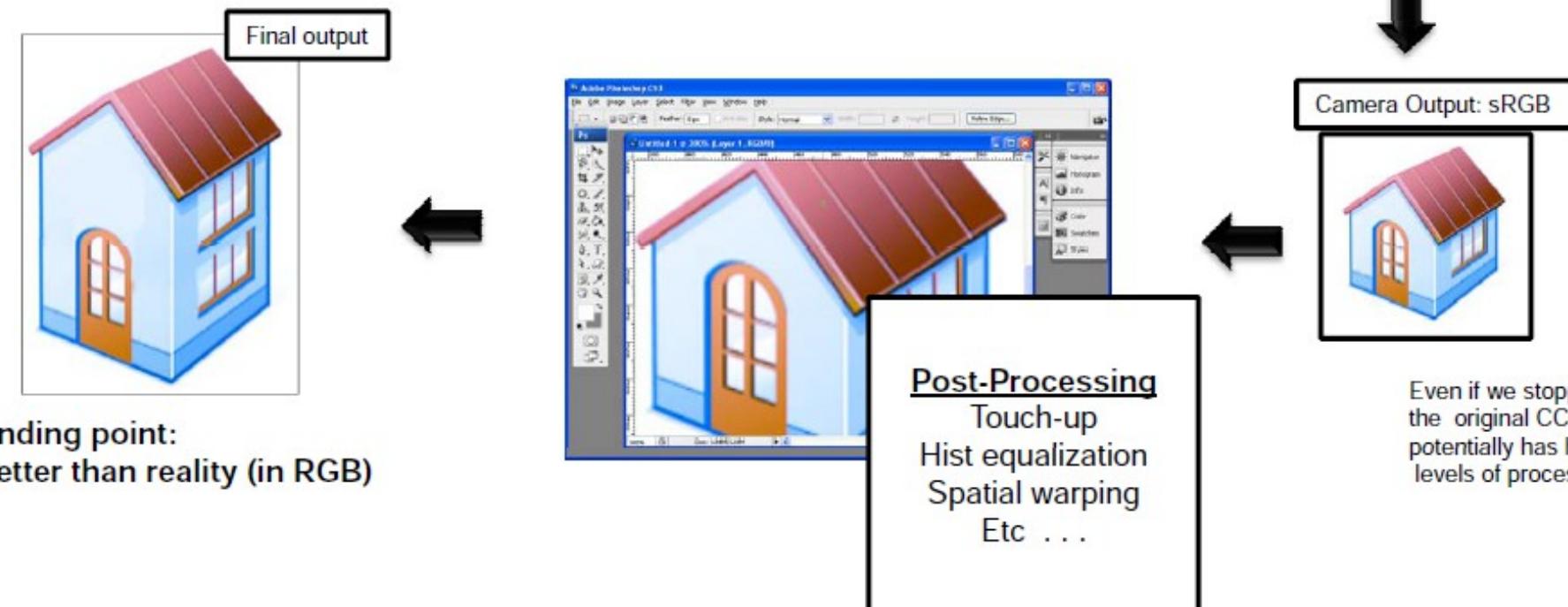
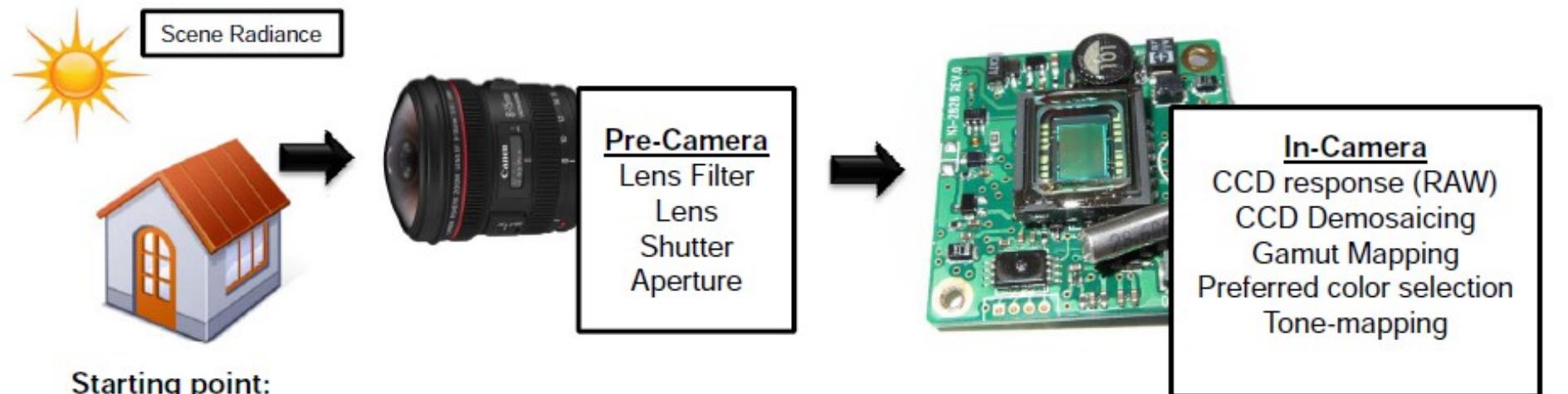


<http://xkcd.com/1014/>

Modern image acquisition pipeline

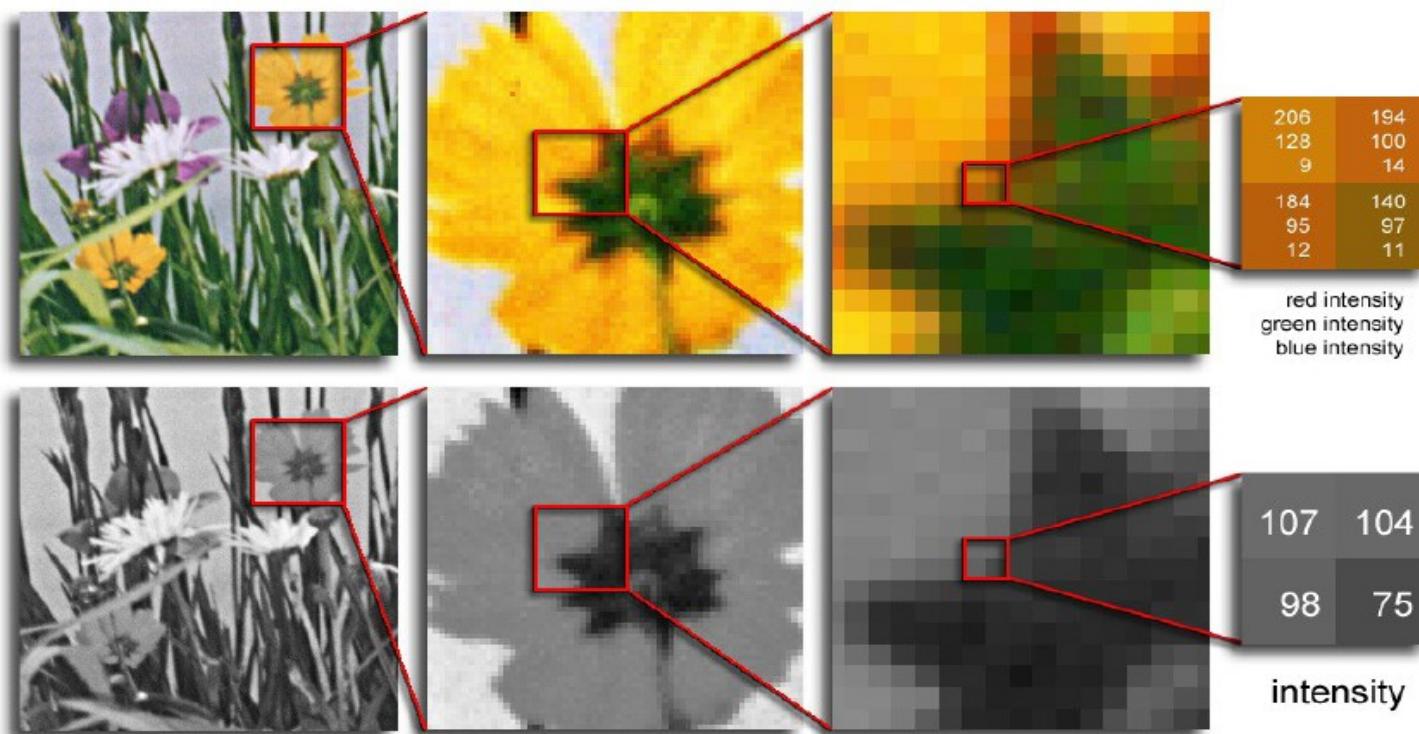


Basics of image processing

Image representation

An image is defined as a 2-dimensional function $f(x,y)$ where x and y are *spatial* coordinates, and the amplitude of f is called *intensity* or *gray level*.

When x,y and f are finite and discrete quantities, we call the image a *digital image*.



Digitalization

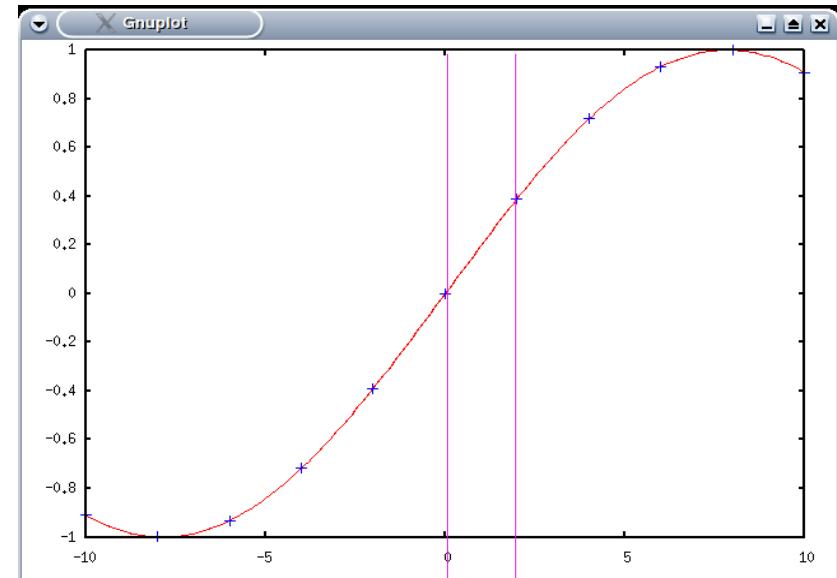
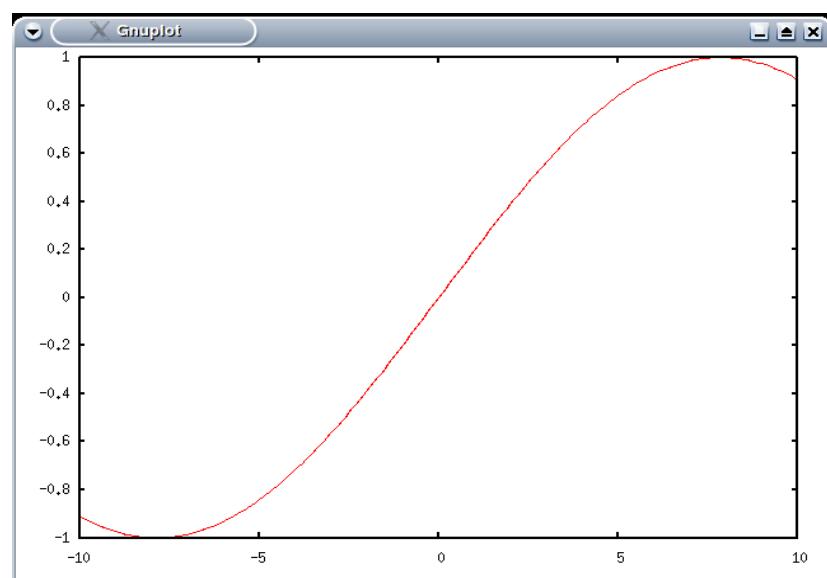
Real data is defined in a continuous domain. However, computers can only handle discrete sets of values.

The way to represent continuous data is to **discretize and quantify**.

Discretization

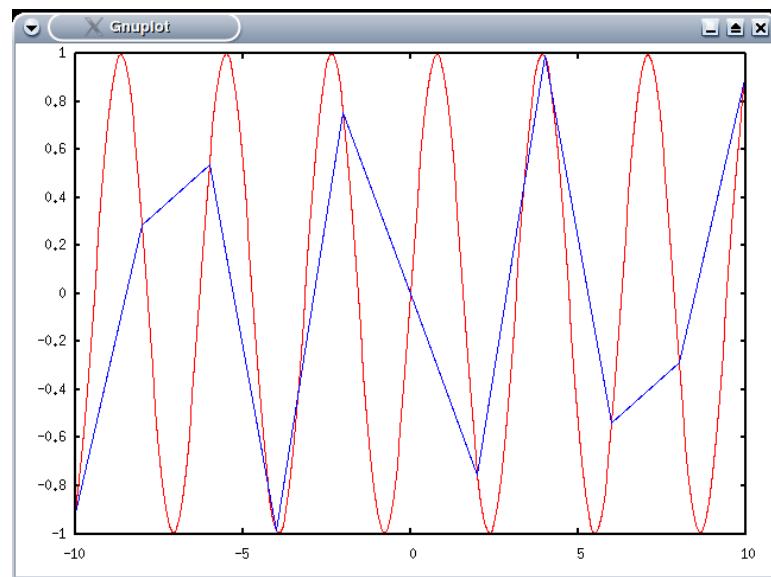
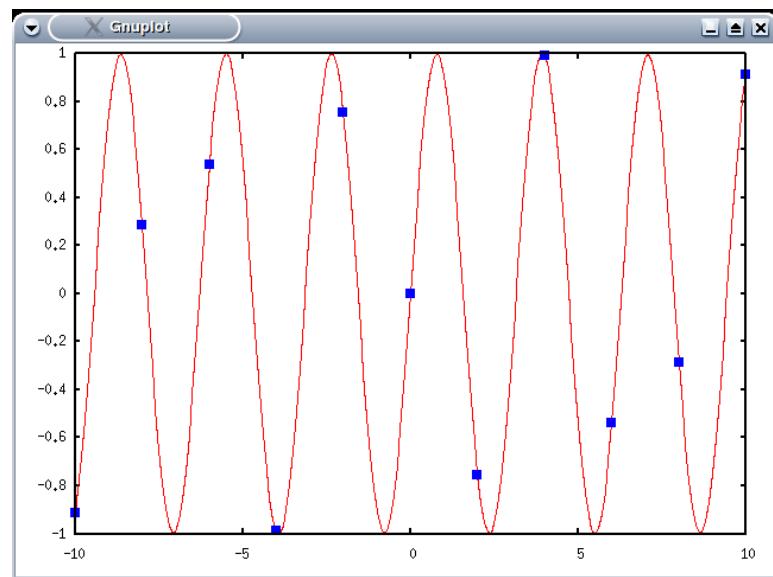
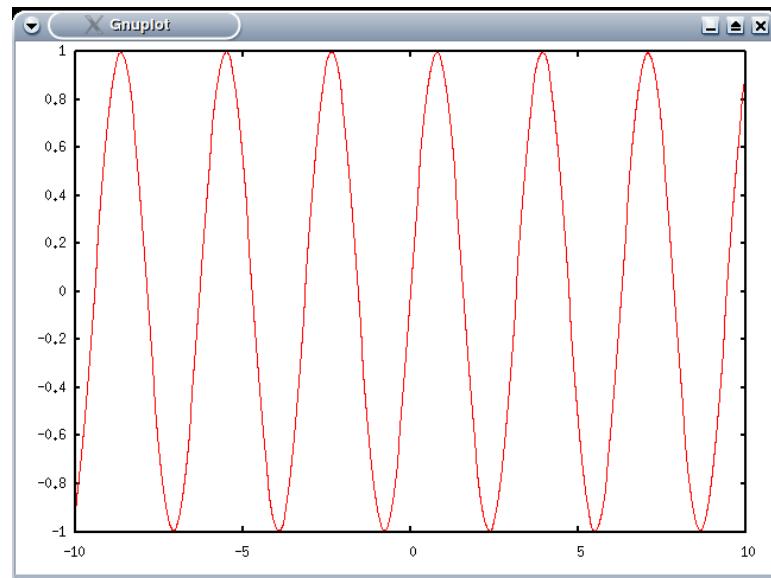
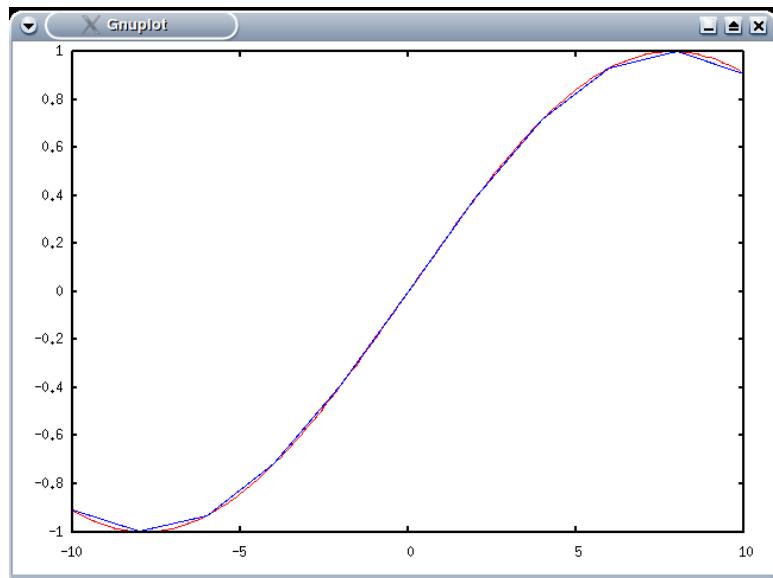
Associated to the concept of sampling and characterized by the sampling frequency or sampling period.

$$x(t) \rightarrow x(n \cdot T_s), \quad \forall n \in \{1 \dots T\}$$



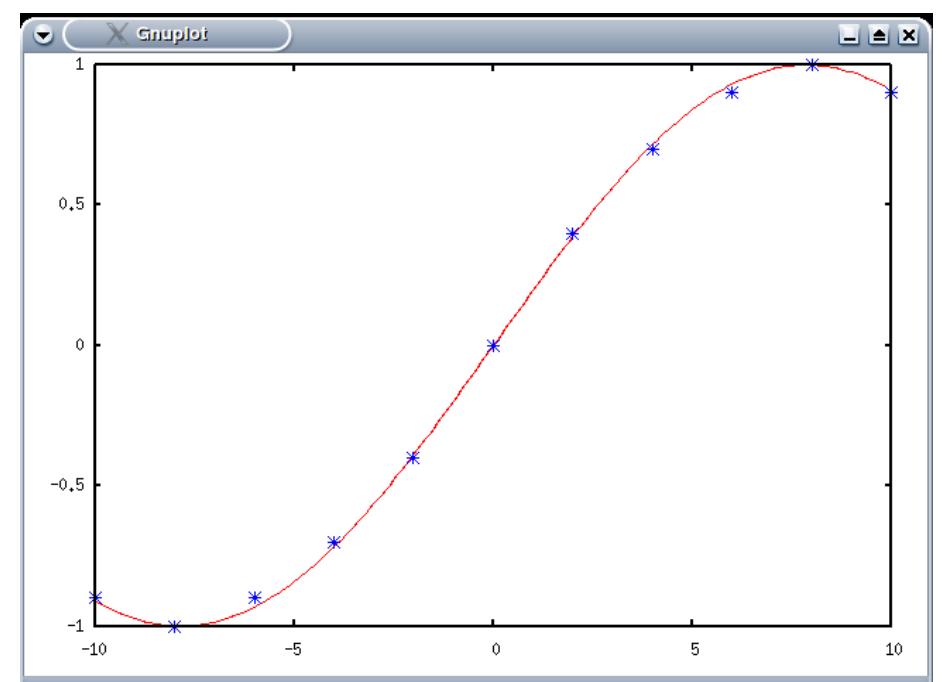
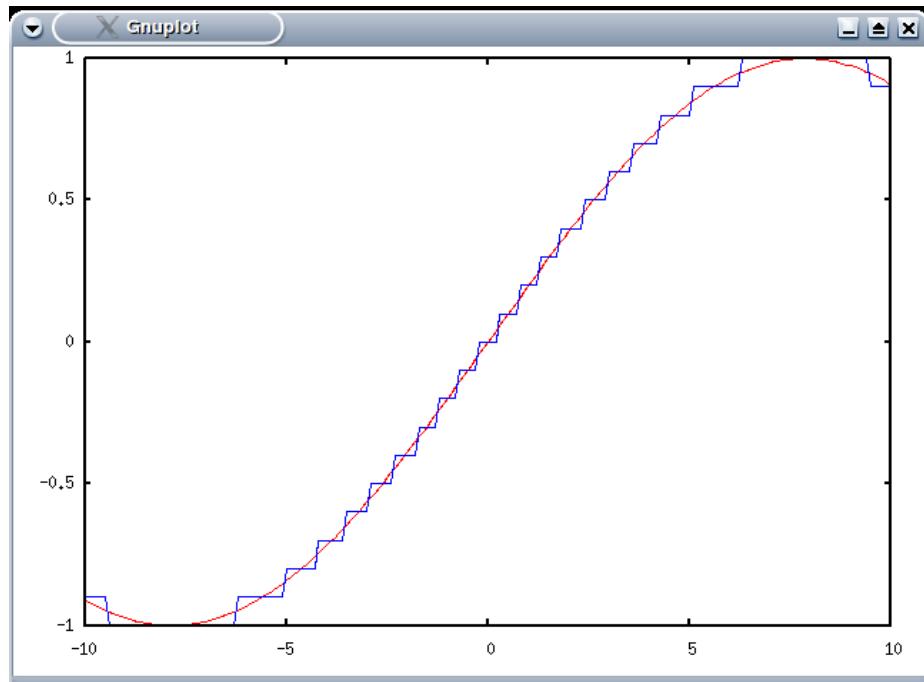
T_s

Reconstruction from sampling



Quantization

Up to this point we have a discrete signal in space/time but the amplitude of the signal is still continuous. The process of discretize the amplitude of the signal is known as **quantization**.



Observe that each level of the quantization correspond to the value stored. In this sense, in an image domain the quantized value is the pixel value.

2D sampling and quantization

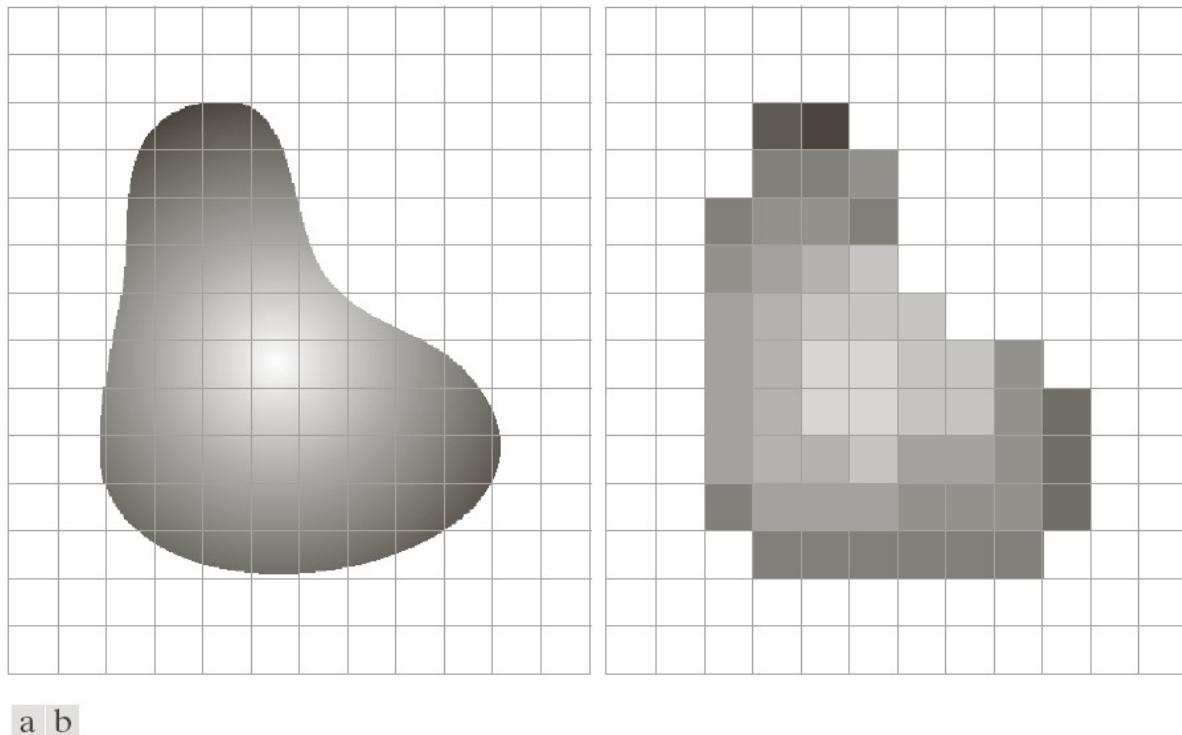


FIGURE 2.17 (a) Continuous image projected onto a sensor array. (b) Result of image sampling and quantization.

The spatial sampling is related to image resolution (e.g. 640x480).
And quantization is related to intensity or color depth (the number of bits per pixel).

Resolution

512x512



64x64



128x128



32x32

Quantization

The number of gray levels associated to b bits is

$$L = 2^b$$



64 levels



16 levels



4 levels



2 levels

Quantization and indexing

The quantization value does not necessarily need to be the extremes of the quantization interval. One may use the mean value or another indexed value. This method is called indexing.



64 levels



16 levels



4 levels



2 levels

Dithering

Dither is an intentionally applied form of noise used to randomize quantization error, preventing large-scale patterns such as "banding" ...



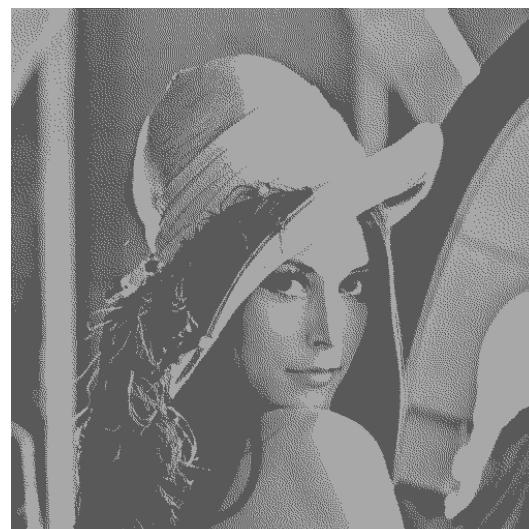
64 levels



16 levels



4 levels



2 levels

Comparison of techniques applied to 2 bits depth images



Brightness

It is a measure of the luminance (global light intensity value) of the image

An estimation of the brightness value:

$$B = \frac{1}{N} \sum_{x,y} f(x, y)$$

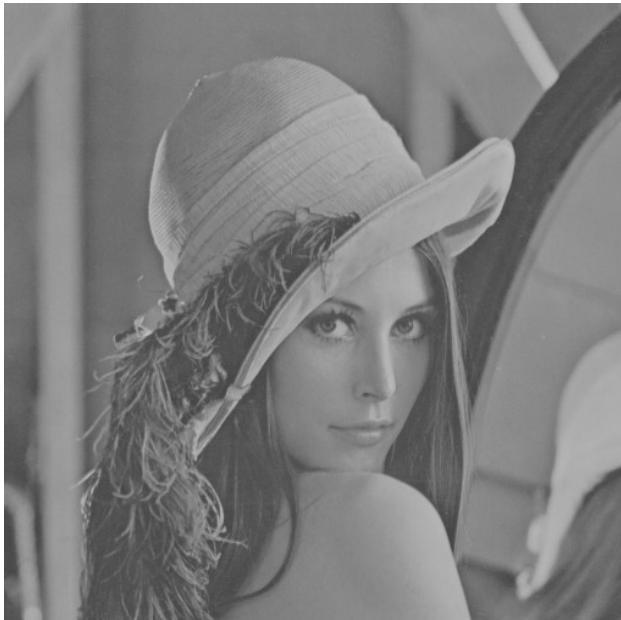


Contrast

Measure of the perceived intensity difference between two regions.

A simple definition of contrast:

$$C = \frac{|f_a - f_b|}{f_a + f_b}$$



Sharpness and Acutance

Acutance is the edge contrast of the image



Sharpness describes the clarity of detail in an image and depends on resolution and acutance



Low res & High ac



High res & Low ac



High res & High ac

Noise

Different types of noise usually categorized by their model:
Additive, multiplicative, impulsive, ...

Examples:

Additive White Gaussian Noise

$$f(x, y) = I(x, y) + n, \quad n \in \mathcal{N}(0, \sigma)$$

Salt and Pepper Noise (Random impulses of extreme values)



Gaussian noise with different sigma

Salt and Pepper

Introduction

The term **spatial domain** refers to the plane of the image.

Spatial methods manipulate pixels themselves.

Basic operation on stand-alone pixels

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

Operations on pixels based on general image information

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

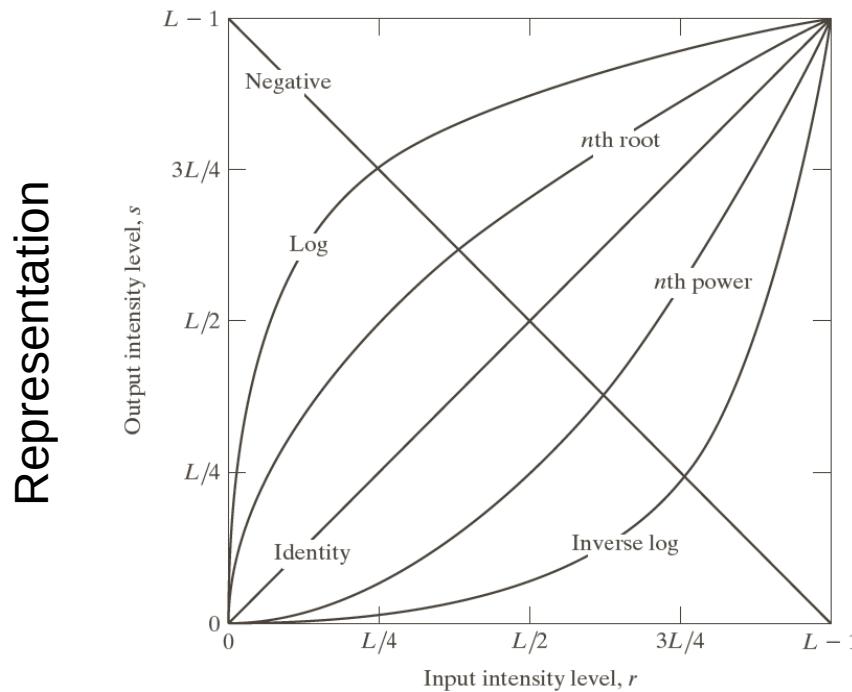
Operations taking into account the neighborhood

$$g(x, y) = T[f(x, y), \mathcal{N}(x, y)]$$

Basic Intensity Transformation Functions

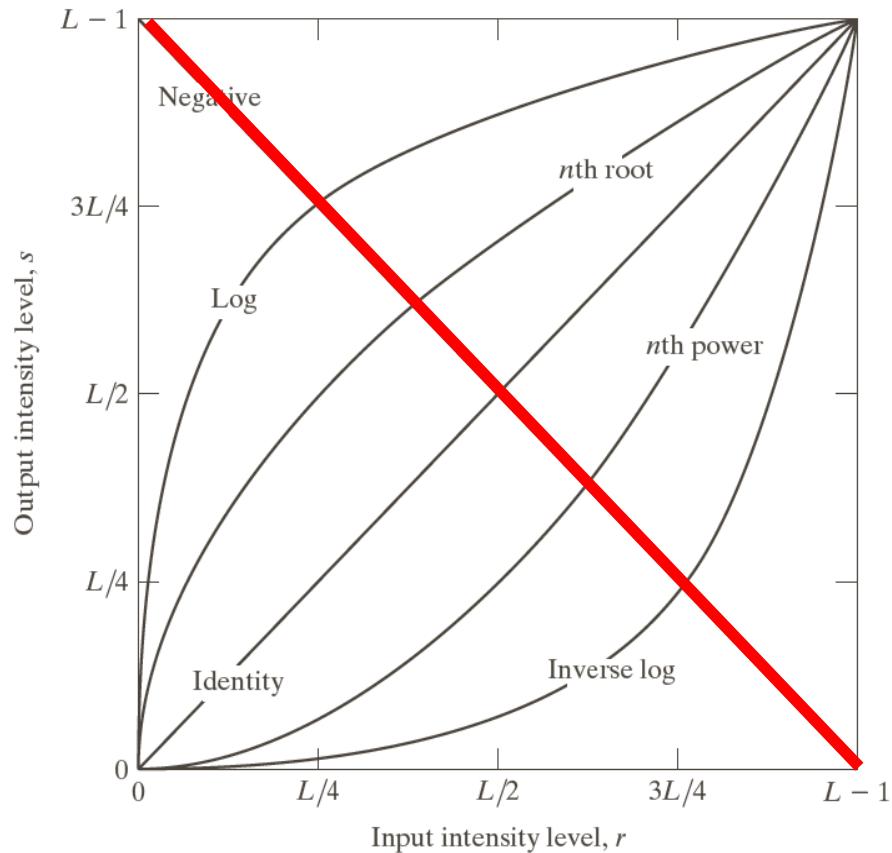
Idea: Replace one intensity value with another.

The simplest transformation is implemented using a Look Up Table (LUT).
For an 8-bit gray image we have to store 256 values.



Negative

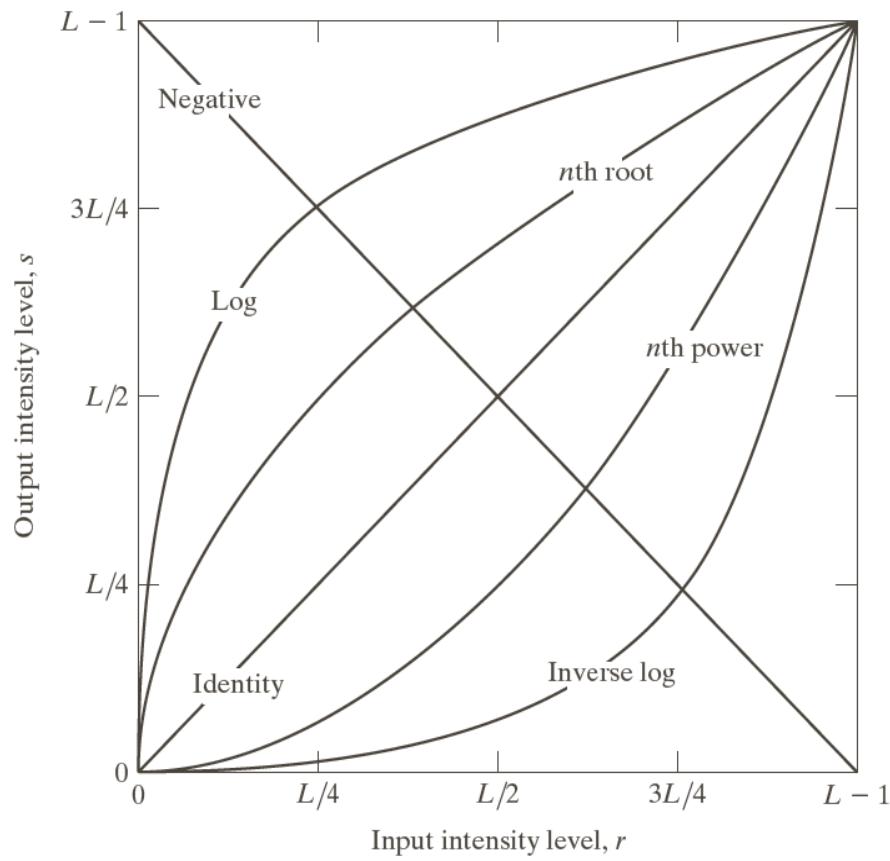
$$g(x, y) = \max(f(x, y)) - f(x, y)$$



Log and Power Transformations

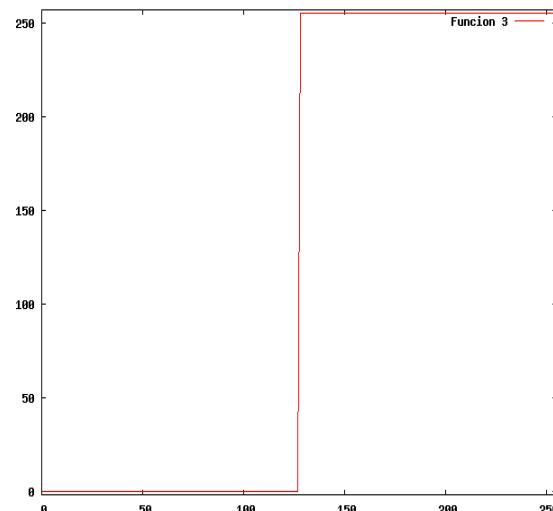
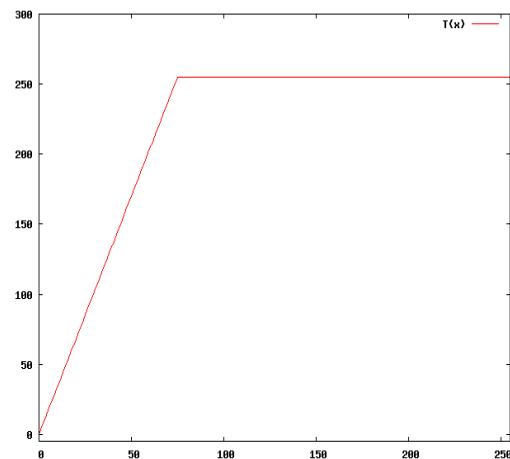
$$g(x, y) = c \log(1 + f(x, y))$$

$$g(x, y) = cf^\gamma(x, y)$$



Effect: compress or expand intensity value range

Piece-wise Transformations



Exercise

Write the transformations or LUTs

Introduction

The term spatial domain refers to the plane of the image.

Spatial methods manipulate pixels themselves.

Basic operation on stand-alone pixels

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

Operations on pixels based on general image information

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

Operations taking into account the neighborhood

$$g(x, y) = T[f(x, y), \mathcal{N}(x, y)]$$

INTUITIVE MECHANIC OF SPATIAL FILTERING

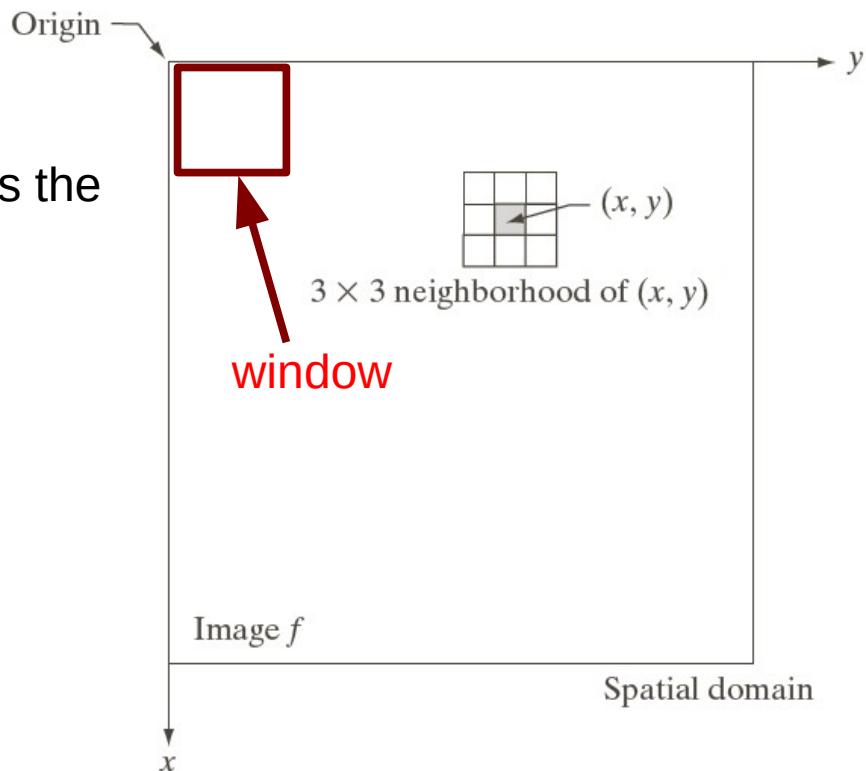
We borrow the name ***filtering*** from the concepts of frequency filtering, where filtering refers to accepting (passing) or rejecting certain frequency components. [More to come in a few weeks]

So, what must we do if we want to process a pixel according to its neighborhood?

Lay a window around the pixel of interest and operate with those values.

And what do we do if we want to process the full image?

Repeat this process for each pixel



INTUITIVE MECHANIC OF SPATIAL FILTERING

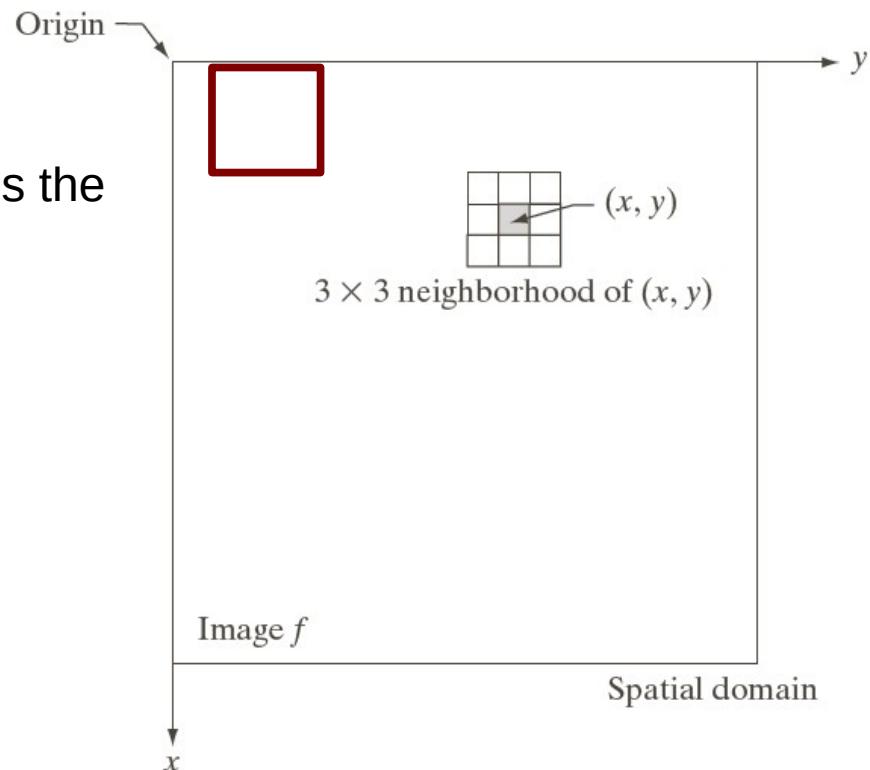
We borrow the name ***filtering*** from the concepts of frequency filtering, where filtering refers to accepting (passing) or rejecting certain frequency components. [More to come in a few weeks]

So, what must we do if we want to process a pixel according to its neighborhood?

Lay a window around the pixel of interest and operate with those values.

And what do we do if we want to process the full image?

Repeat this process for each pixel



INTUITIVE MECHANIC OF SPATIAL FILTERING

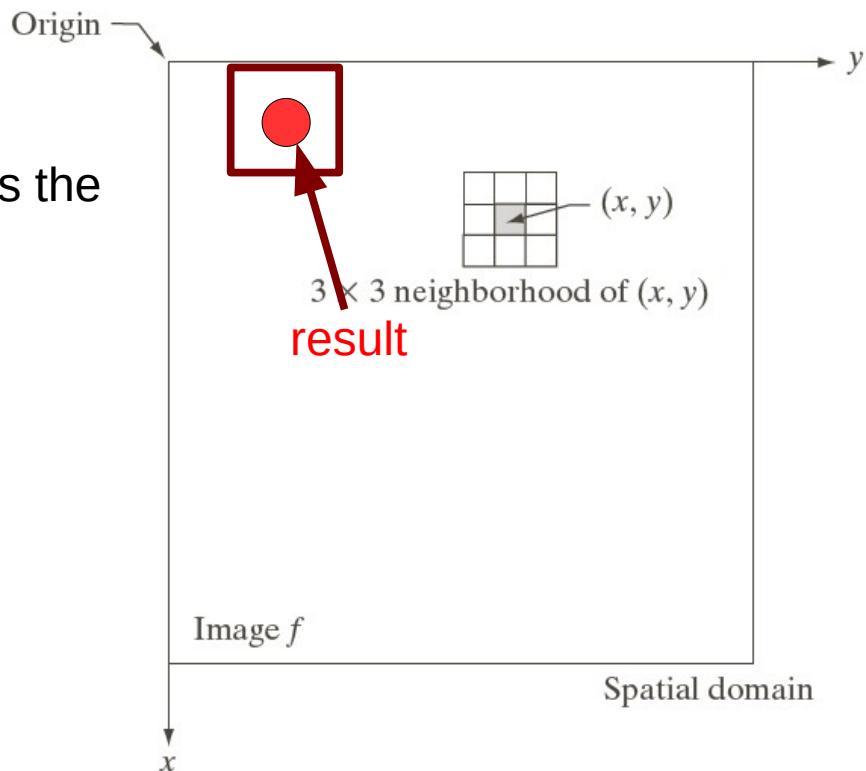
We borrow the name ***filtering*** from the concepts of frequency filtering, where filtering refers to accepting (passing) or rejecting certain frequency components. [More to come in a few weeks]

So, what must we do if we want to process a pixel according to its neighborhood?

Lay a window around the pixel of interest and operate with those values.

And what do we do if we want to process the full image?

Repeat this process for each pixel



What for?

- Enhance images
 - Denoise
 - Resize
 - Increase contrast

- Extract features
 - Texture
 - Edges
 - Salient points

- Detect patterns
 - Template matching

A SIMPLE EXAMPLE

We want to smooth an image. For this task we may use the mean value in a 3x3 window.

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

$$\begin{aligned} & \frac{1}{9}(f(x-1, y-1) + f(x-1, y) + f(x-1, y+1) + \dots \\ & + f(x, y-1) + f(x, y) + f(x, y+1) + \dots \\ & + f(x+1, y-1) + f(x+1, y) + f(x+1, y+1)) \end{aligned}$$

A SIMPLE EXAMPLE

We want to smooth an image. For this task we may use the mean value in a 3x3 window.

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

OBSERVE THAT THE SAME RESULT CAN BE OBTAINED USING
A WINDOW WITH WEIGHTS AND ADDING THE ELEMENT
MULTIPLICATION

$w(-1,-1)$	$w(-1,0)$	$w(-1,1)$
$w(0,-1)$	$w(0,0)$	$w(0,1)$
$w(1,-1)$	$w(1,0)$	$w(1,1)$

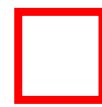
} MASK

$$\frac{1}{9} \sum_{i,j \in \{-1,0,1\}} w(i,j) f(x+i, y+j)$$

EXAMPLE: BOX FILTER

$$\frac{1}{9} \begin{matrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{matrix} \quad w(i, j)$$

$$h(x, y)$$



$$f(x, y)$$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$$h(x, y) = \sum_{i,j \in \{-1,0,1\}} w(i, j) f(x + i, y + j)$$

EXAMPLE: BOX FILTER

$$\frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \quad w(i, j)$$

$$f(x, y)$$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$$h(x, y)$$

$$0 \quad 10$$

$$h(x, y) = \sum_{i,j \in \{-1,0,1\}} w(i, j) f(x + i, y + j)$$

EXAMPLE: BOX FILTER

$$\frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \quad w(i, j)$$

$$f(x, y)$$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$$h(x, y)$$

$$0 \quad 10 \quad \boxed{20}$$

$$h(x, y) = \sum_{i,j \in \{-1,0,1\}} w(i, j) f(x + i, y + j)$$

EXAMPLE: BOX FILTER

$$\frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \quad w(i, j)$$

$$f(x, y)$$

0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0
0	0	0	90	90	90	90	90	0
0	0	0	90	90	90	90	90	0
0	0	0	90	0	90	90	90	0
0	0	0	90	90	90	90	90	0
0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0

$$h(x, y)$$



$$h(x, y) = \sum_{i,j \in \{-1,0,1\}} w(i, j) f(x + i, y + j)$$

EXAMPLE: BOX FILTER

$$\frac{1}{9} \begin{matrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{matrix} \quad w(i, j)$$

$f(x, y)$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$h(x, y)$

0 10 20 30 30

?

$$h(x, y) = \sum_{i,j \in \{-1,0,1\}} w(i, j) f(x + i, y + j)$$

EXAMPLE: BOX FILTER

$$\frac{1}{9} \begin{matrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{matrix} \quad w(i, j)$$

$$f(x, y)$$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$$h(x, y)$$

0	10	20	30	30	30	20	10
0	20	40	60	60	60	40	20
0	30	60	90	90	90	60	30
0	30	50	80	80	90	60	30
0	30	50	80	80	90	60	30
0	20	30	50	50	60	40	20
10	20	30	30	30	30	20	10
10	10	10	0	0	0	0	0

$$h(x, y) = \sum_{i,j \in \{-1,0,1\}} w(i, j) f(x + i, y + j)$$

EXAMPLE: BOX FILTER

What does it do?

- Replaces each pixel with an average of its neighborhood
- Achieve smoothing effect
(remove sharp features)

$$\frac{1}{9} \begin{array}{|c|c|c|} \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline \end{array} w(i, j)$$



PRACTICE WITH FILTERS



Original

0	0	0
0	1	0
0	0	0

?

PRACTICE WITH FILTERS



Original

0	0	0
0	1	0
0	0	0



Remains the same

PRACTICE WITH FILTERS



Original

0	0	0
0	0	1
0	0	0

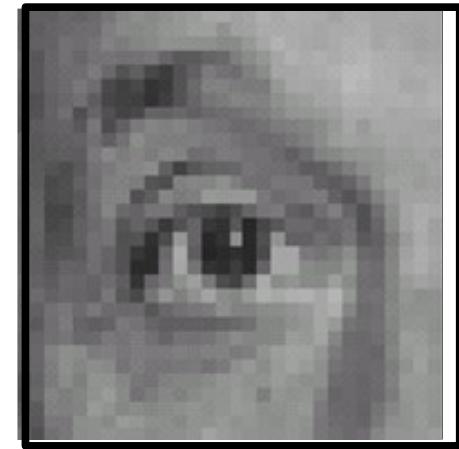
?

PRACTICE WITH FILTERS



Original

0	0	0
0	0	1
0	0	0



Shift 1 pixel to the left

PRACTICE WITH FILTERS



Original

0	0	0
0	2	0
0	0	0

- 1/9

1	1	1
1	1	1
1	1	1

?

PRACTICE WITH FILTERS



Original

0	0	0
0	2	0
0	0	0

- 1/9

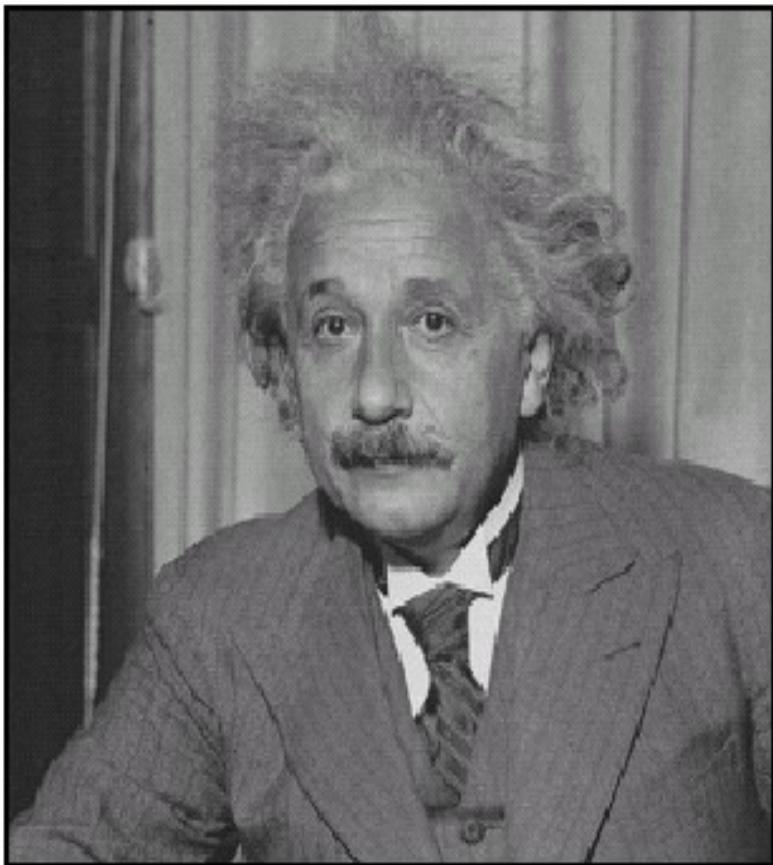
1	1	1
1	1	1
1	1	1



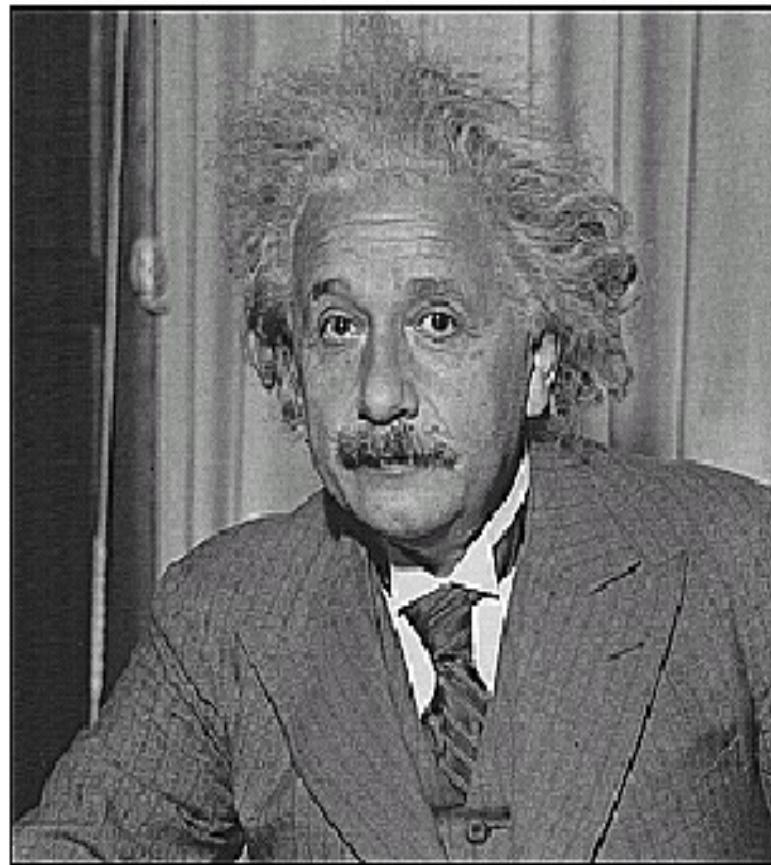
Sharpening filter

- Accentuates differences with local average

SHARPENING

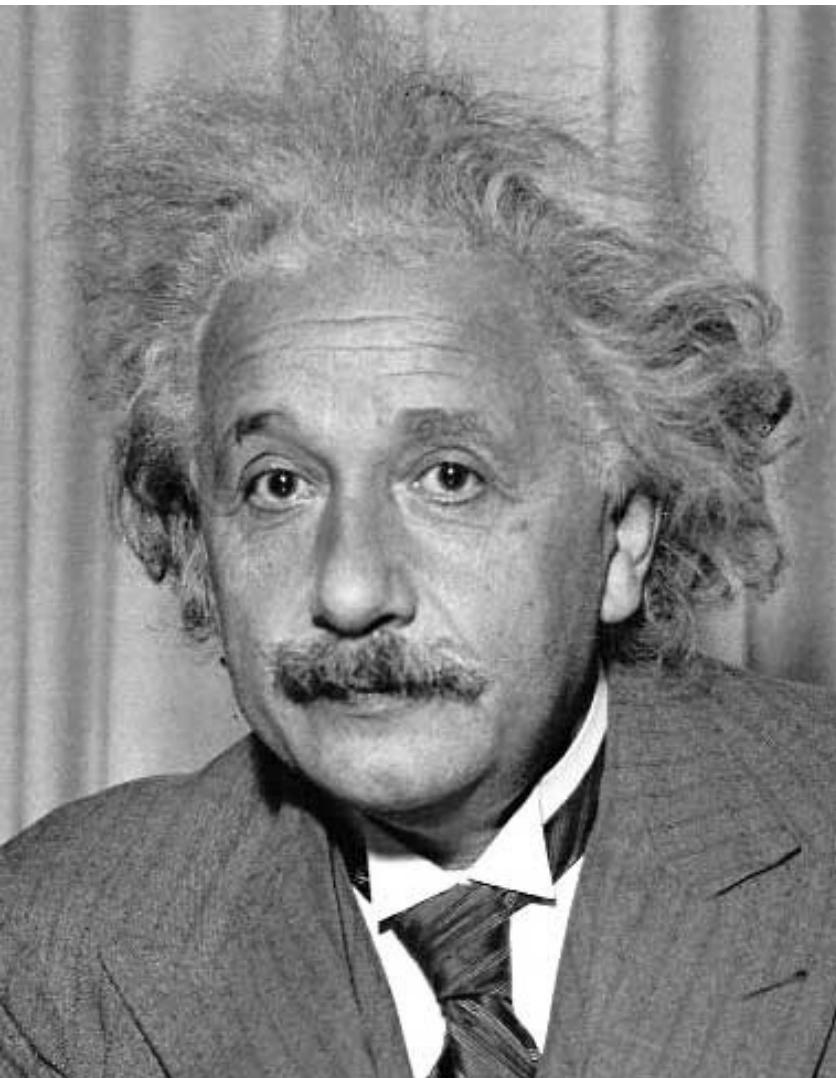


before



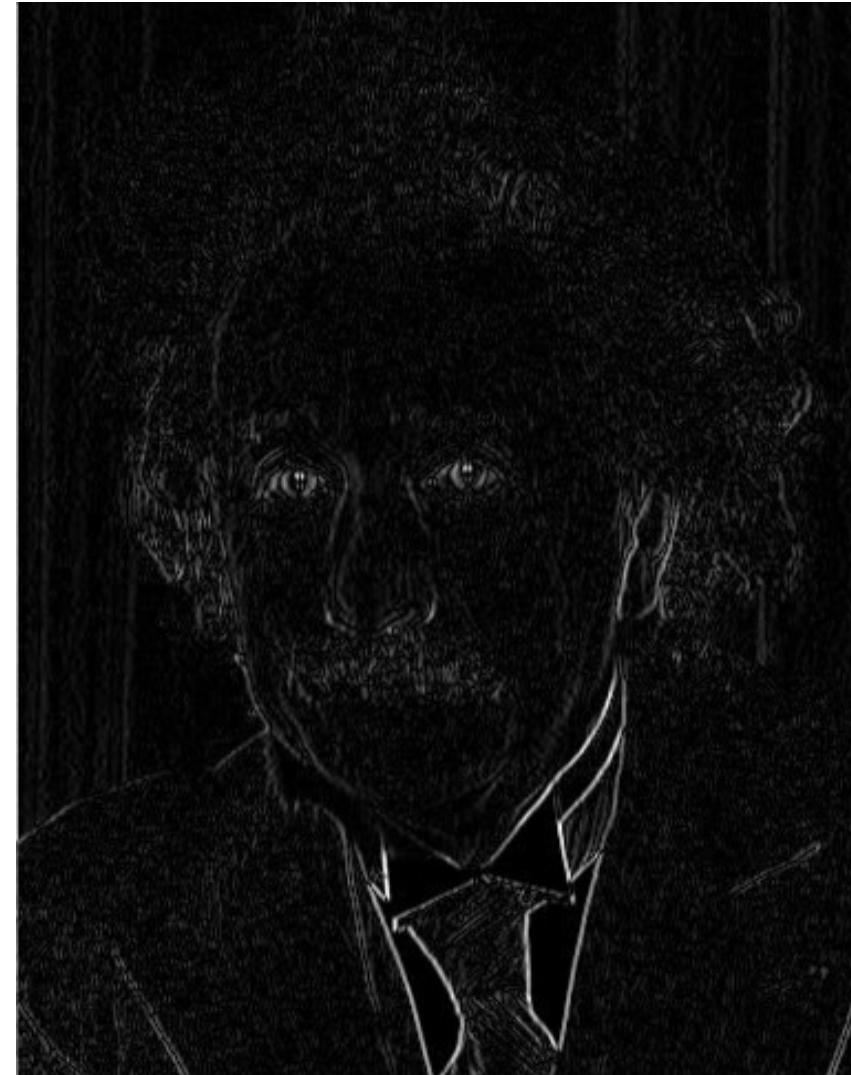
after

PRACTICE WITH FILTERS



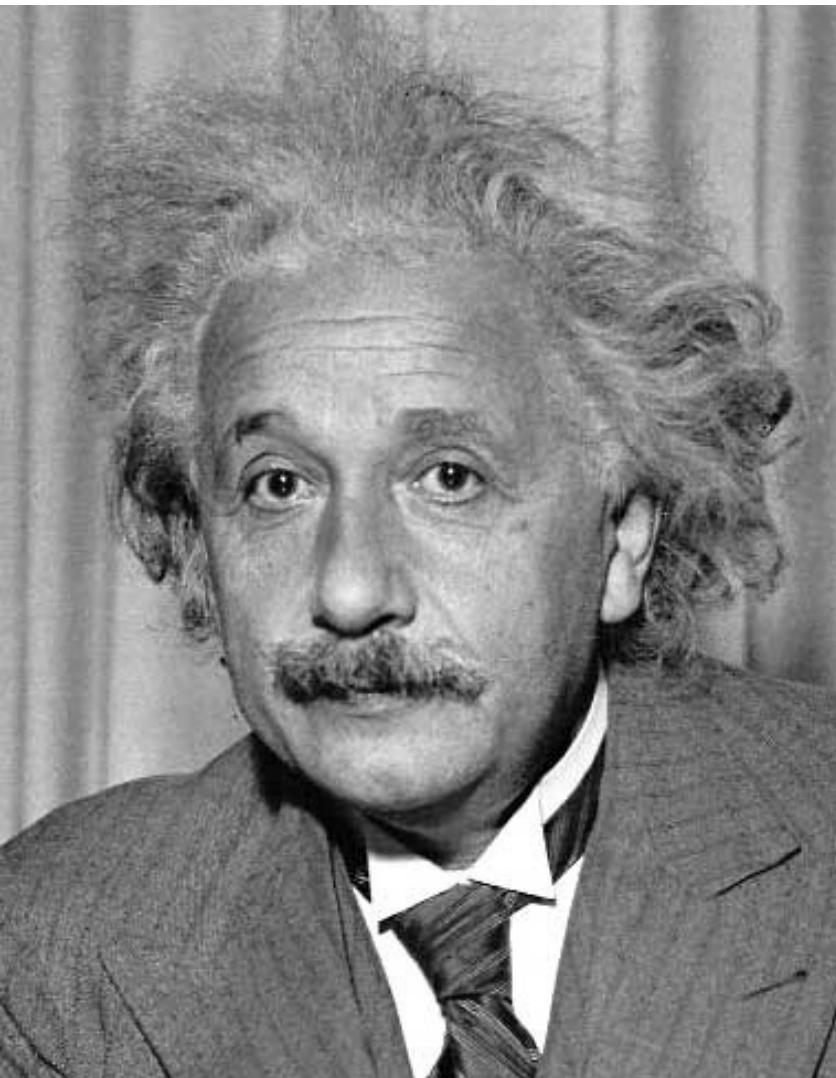
1	0	-1
2	0	-2
1	0	-1

Sobel



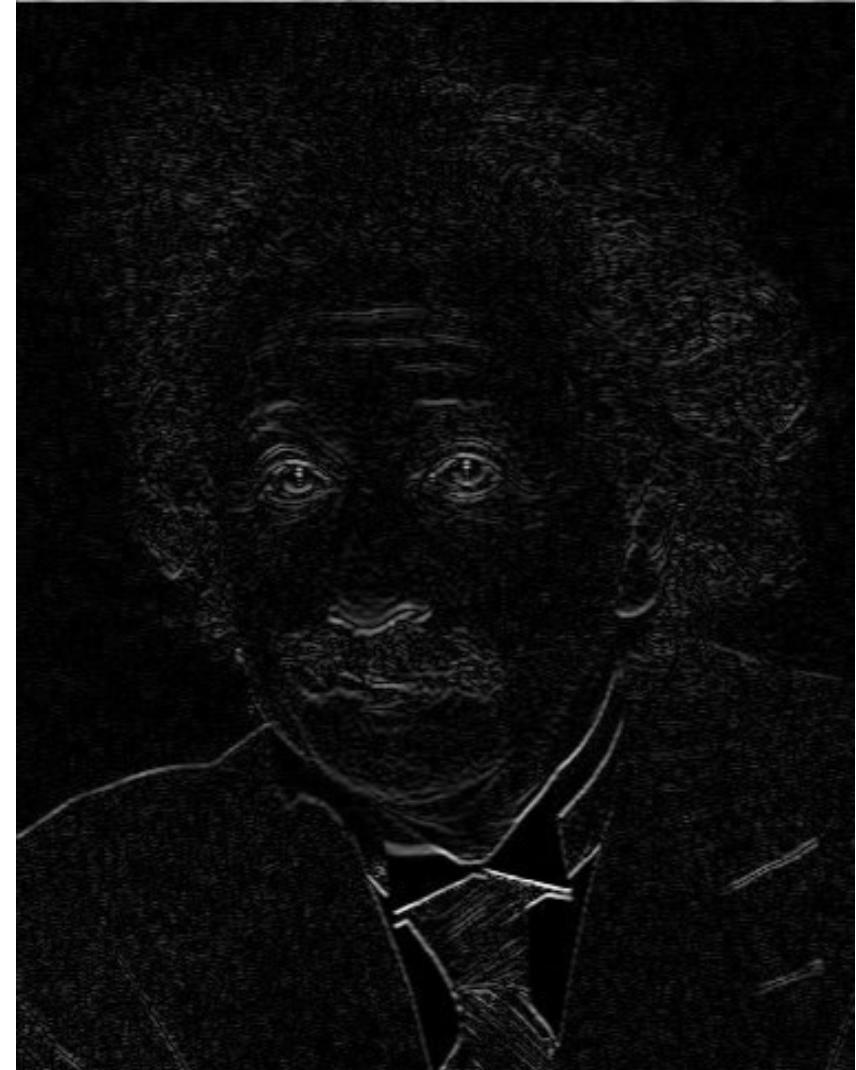
Vertical Edge
(absolute value)

PRACTICE WITH FILTERS



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

Sobel



Horizontal Edge
(absolute value)

NOW... FOR THE REAL DEAL

This process of sliding window in general is called **correlation**.

If we multiply the same function it is called **auto-correlation**.

If the functions are different it is also known as **cross-correlation**.

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

The importance of this operation is that it gives a maximum when both functions are the most similar. Thus, the more similar the functions are, the higher the value of the correlation is.

The process of reversing one of the signals and cross-correlate them is the well known **CONVOLUTION**. This is defined as follows,

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

KEY PROPERTIES OF LINEAR FILTERS

Linearity:

$$\text{filter}(f_1 + f_2) = \text{filter}(f_1) + \text{filter}(f_2)$$

Shift invariance: same behavior regardless of pixel location

$$\text{filter}(\text{shift}(f)) = \text{shift}(\text{filter}(f))$$

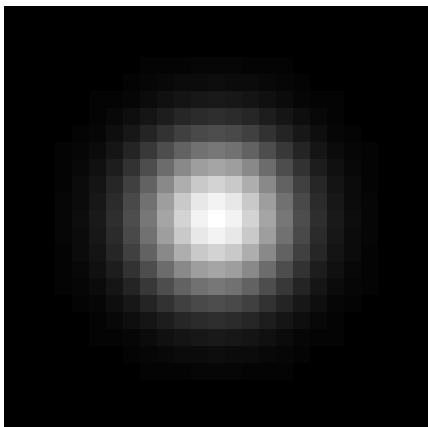
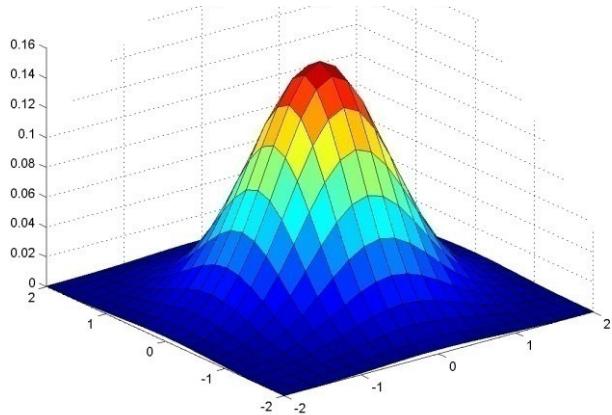
Any linear, shift-invariant operator can be represented as a convolution

OTHER PROPERTIES OF LINEAR FILTERS

- Commutative: $a * b = b * a$
 - Conceptually no difference between filter and signal
- Associative: $a * (b * c) = (a * b) * c$
 - Often apply several filters one after another: $((a * b1) * b2) * b3$
 - This is equivalent to applying one filter: $a * (b1 * b2 * b3)$
- Distributes over addition: $a * (b + c) = (a * b) + (a * c)$
- Scalars factor out: $ka * b = a * kb = k (a * b)$
- Identity: unit impulse $e = [0, 0, 1, 0, 0]$,
 $a * e = a$

AN IMPORTANT FILTER: GAUSSIAN

- Weight contributions of neighboring pixels by nearness

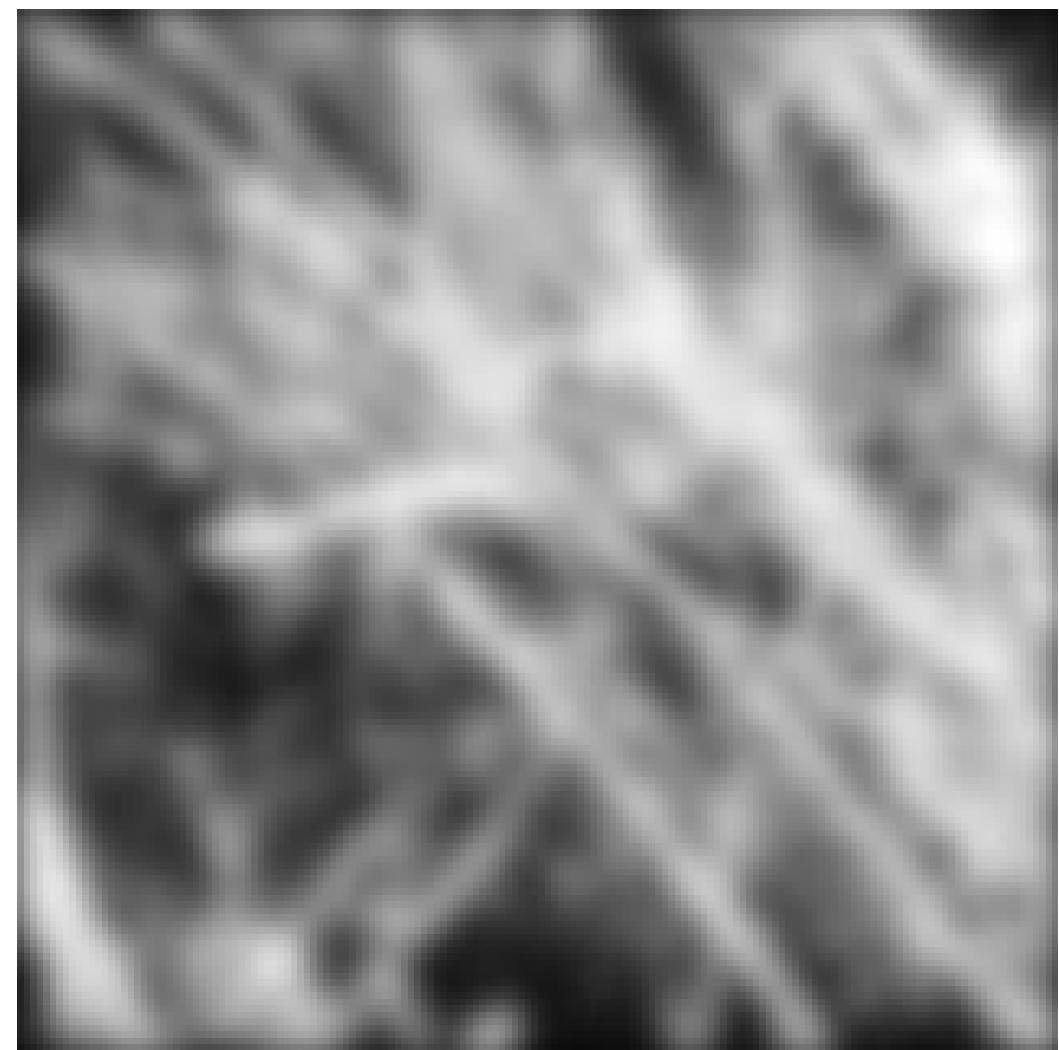


0.003	0.013	0.022	0.013	0.003
0.013	0.059	0.097	0.059	0.013
0.022	0.097	0.159	0.097	0.022
0.013	0.059	0.097	0.059	0.013
0.003	0.013	0.022	0.013	0.003

$5 \times 5, \sigma = 1$

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

SMOOTHING WITH A GAUSSIAN FILTER



SMOOTHING WITH A BOX FILTER



GAUSSIAN FILTER PROPERTIES

- Remove “high-frequency” components from the image (low-pass filter)
 - Images become more smooth
- Convolution with self is another Gaussian
 - So can smooth with small-width kernel, repeat, and get same result as larger-width kernel would have
 - Convolving two times with Gaussian kernel of width σ is same as convolving once with kernel of width $\sigma\sqrt{2}$
- *Separable* kernel
 - Factors into product of two 1D Gaussians

GAUSSIAN FILTER PROPERTIES

$$\begin{aligned} G_\sigma(x, y) &= \frac{1}{2\pi\sigma^2} \exp^{-\frac{x^2 + y^2}{2\sigma^2}} \\ &= \left(\frac{1}{\sqrt{2\pi}\sigma} \exp^{-\frac{x^2}{2\sigma^2}} \right) \left(\frac{1}{\sqrt{2\pi}\sigma} \exp^{-\frac{y^2}{2\sigma^2}} \right) \end{aligned}$$

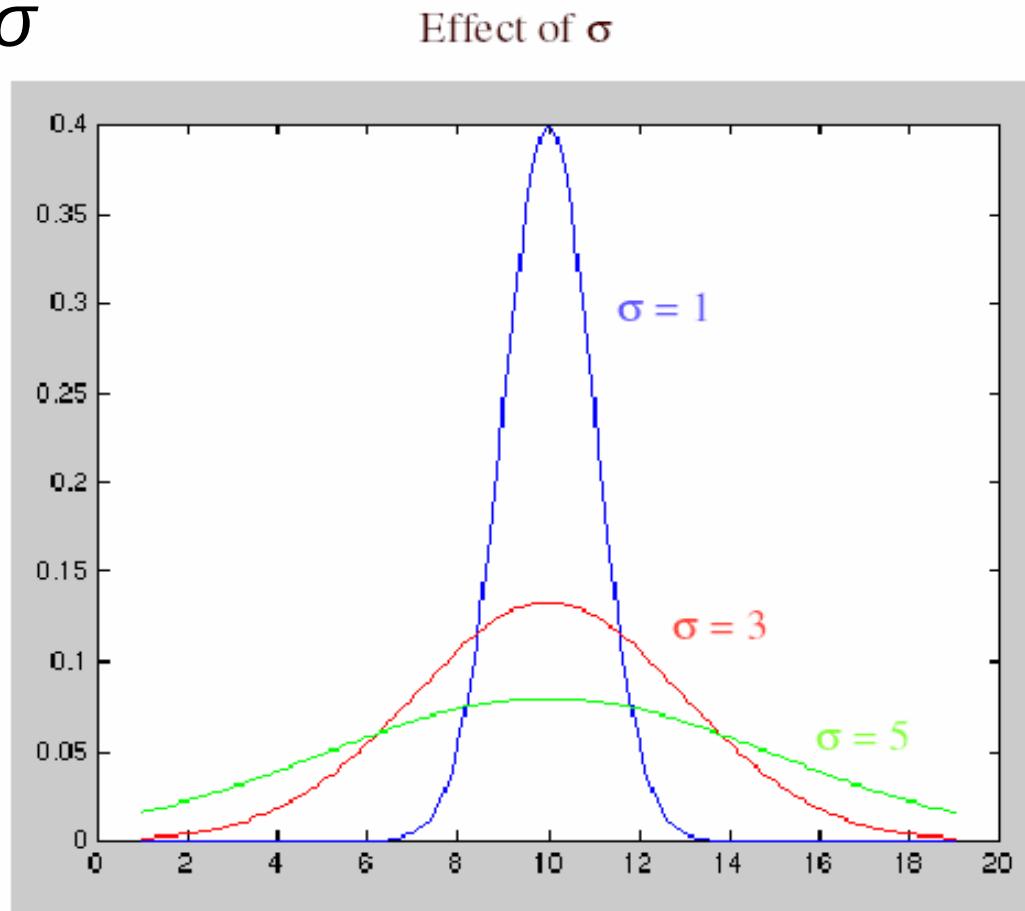
The 2D Gaussian can be expressed as the product of two functions, one a function of x and the other a function of y

In this case, the two functions are the (identical) 1D Gaussian

PRACTICAL MATTERS

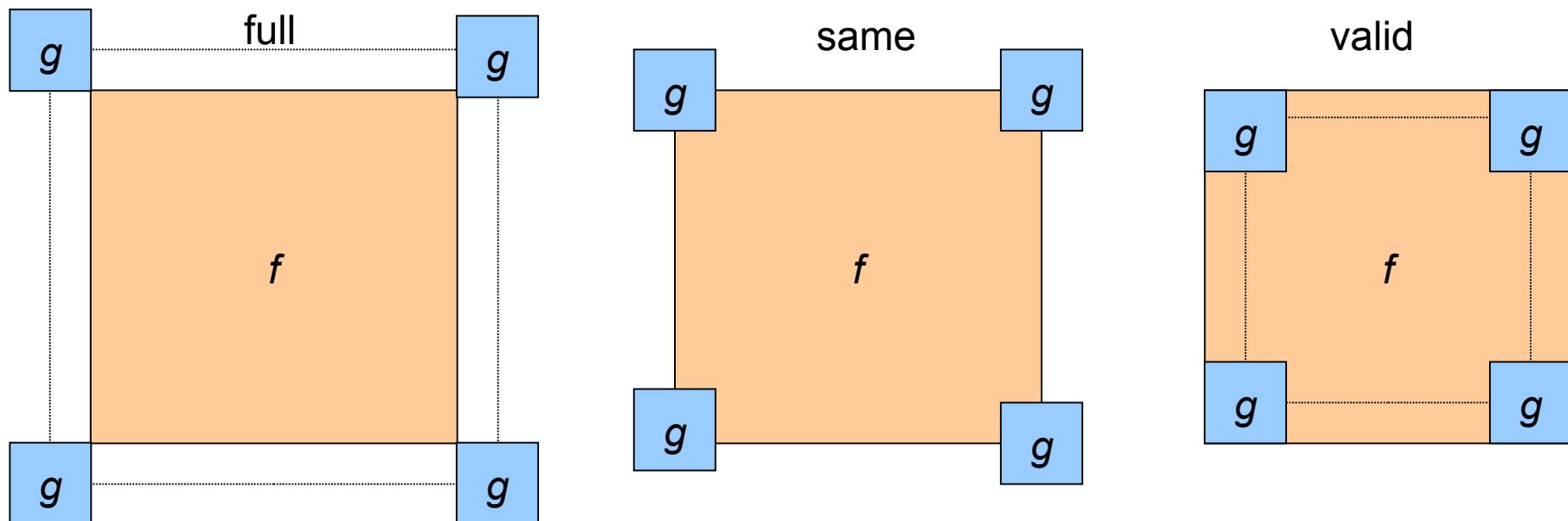
How big should the filter be?

- Values at edges should be near zero
- Rule of thumb for Gaussian: set filter half-width to about 3σ



PRACTICAL MATTERS

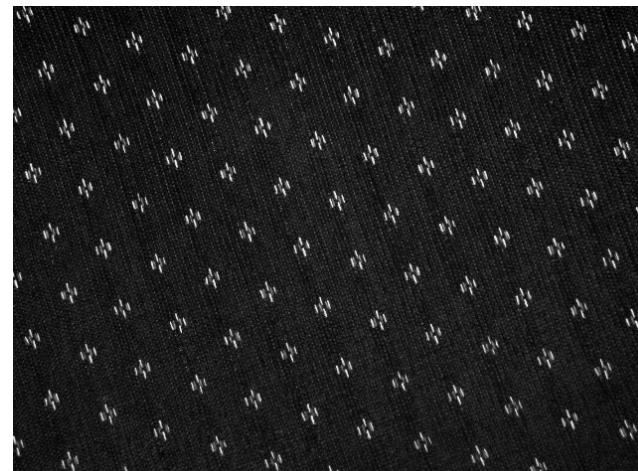
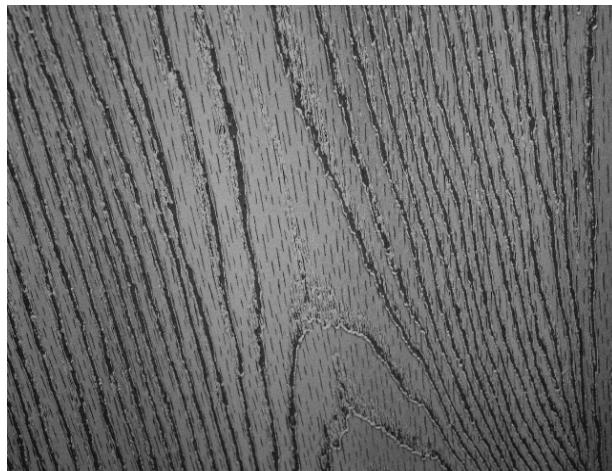
- What is the size of the output?
- MATLAB: `filter2(g, f, shape)`
 - `shape = 'full'`: output size is sum of sizes of `f` and `g`
 - `shape = 'same'`: output size is same as `f`
 - `shape = 'valid'`: output size is difference of sizes of `f` and `g`



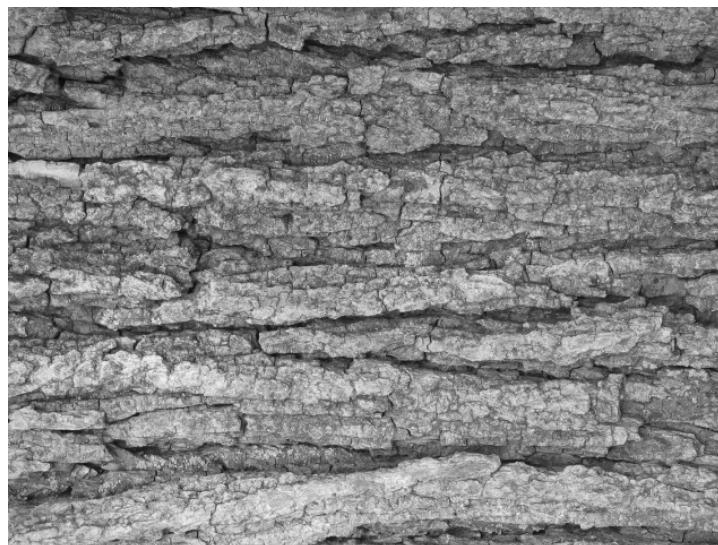
APPLICATION OF FILTERS: TEXTURES



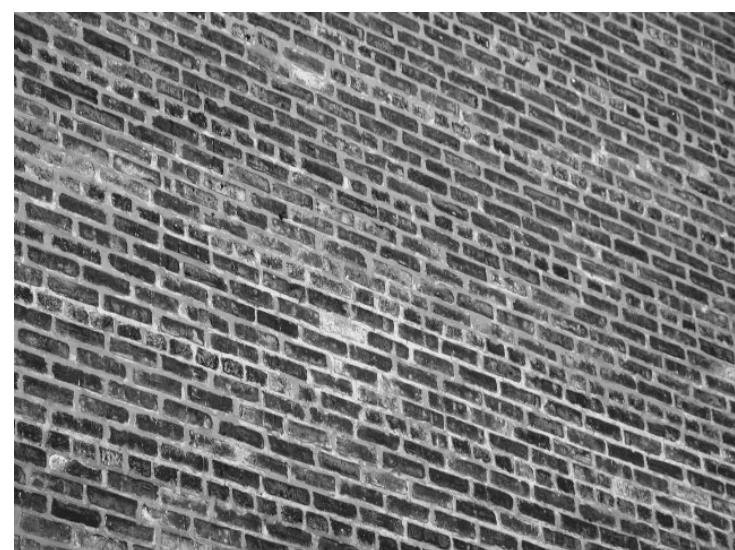
TEXTURES AND MATERIALS



TEXTURES AND ORIENTATIONS



TEXTURES AND SCALES



WHAT IS A TEXTURE?

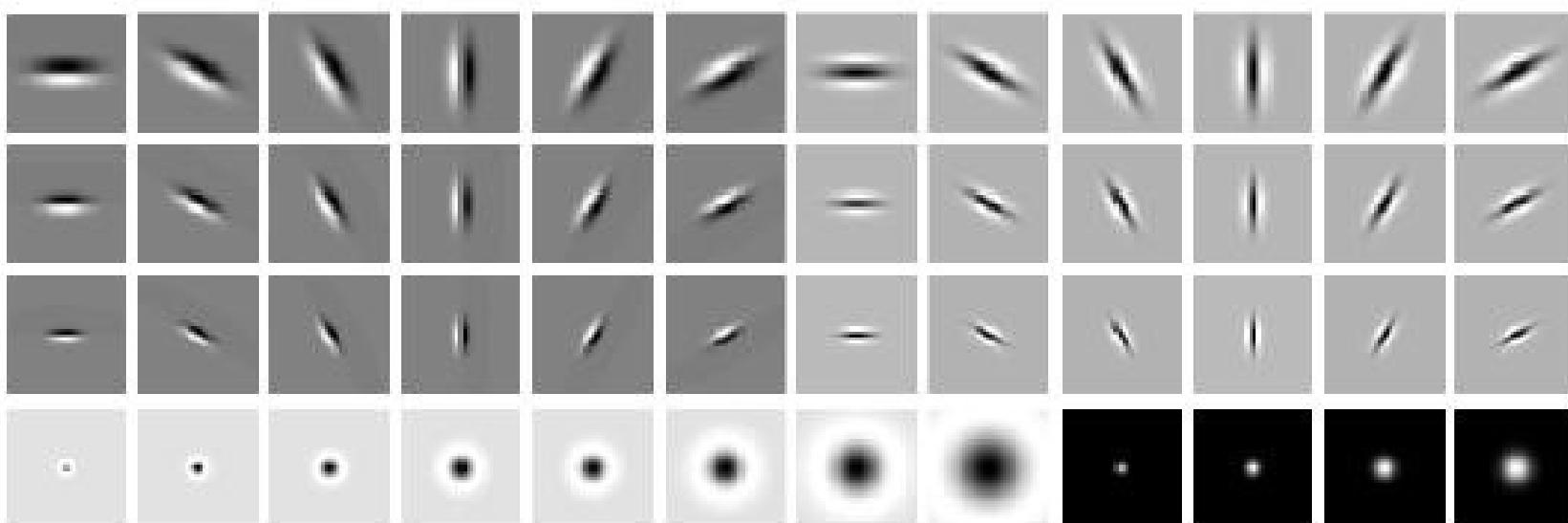
Regular or stochastic patterns caused by bumps, grooves, and/or markings

HOW CAN WE REPRESENT TEXTURE?

Compute responses of blobs and edges at various orientations and scales

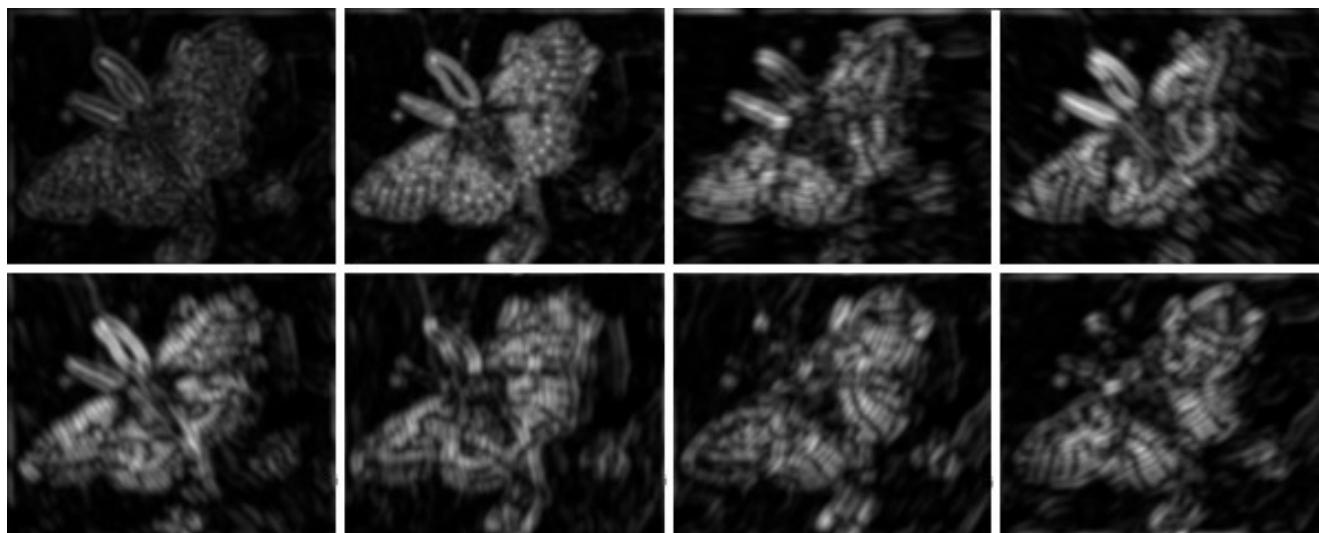
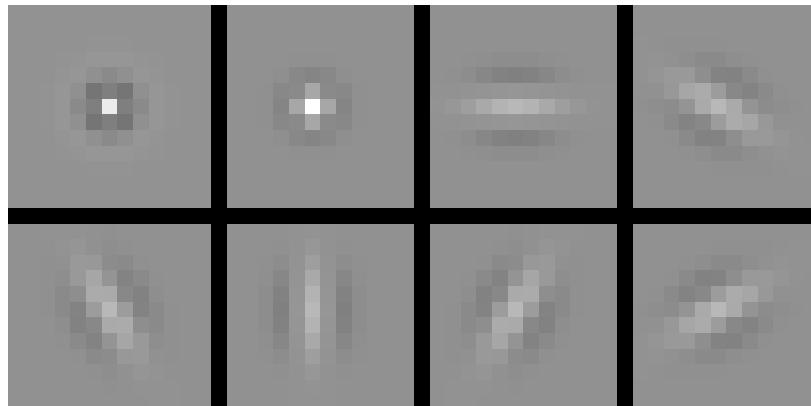
OVERCOMPLETE REPRESENTATION: FILTER BANKS

LM Filter Bank

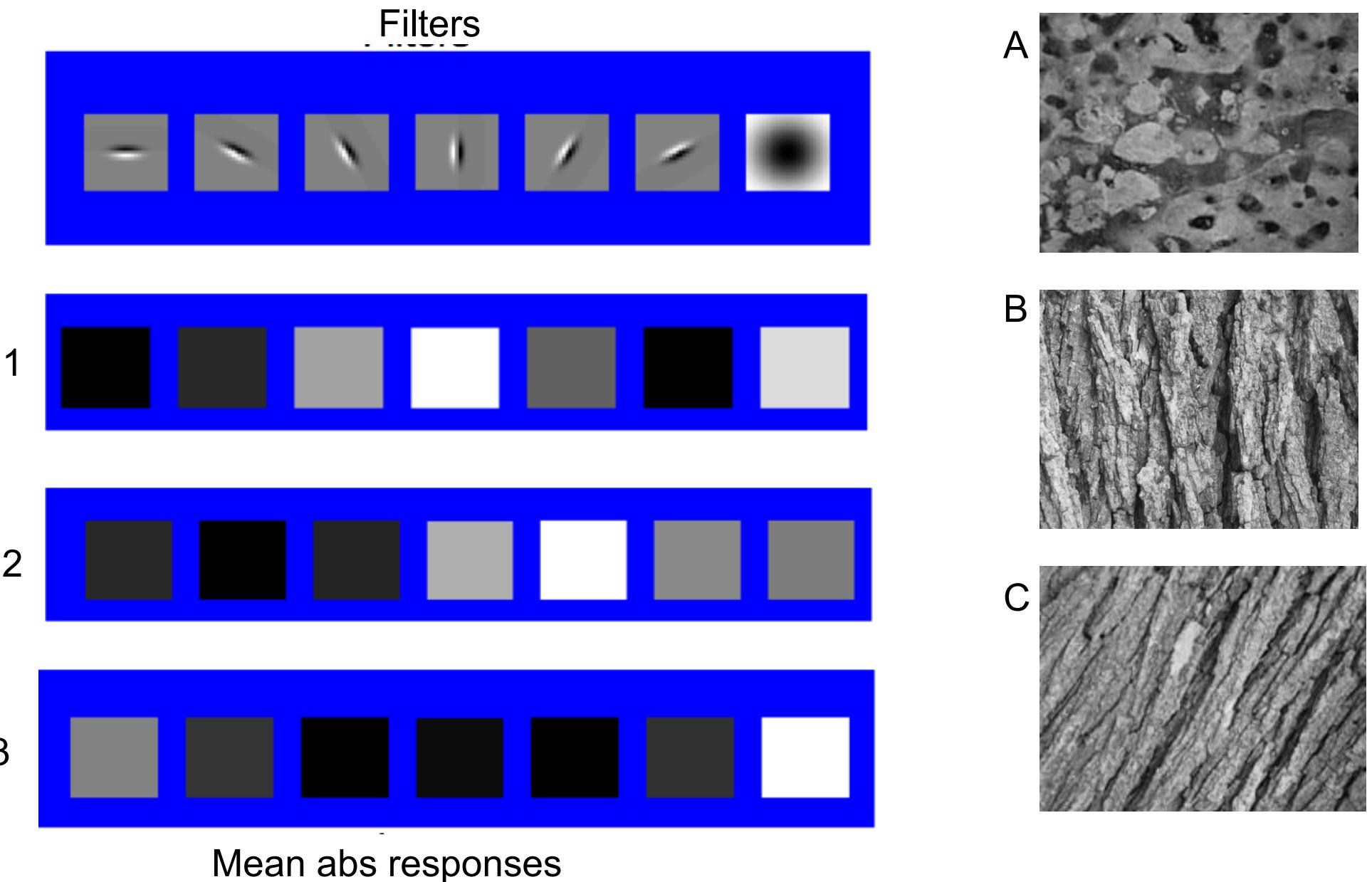


HOW TO USE IT...

- Process image with each filter and keep responses (or squared/abs responses)

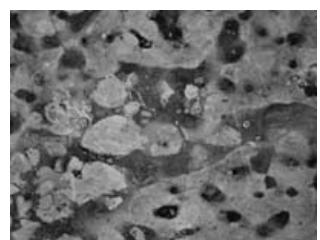
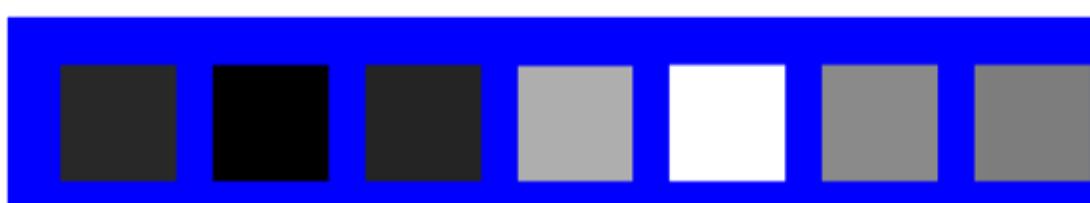
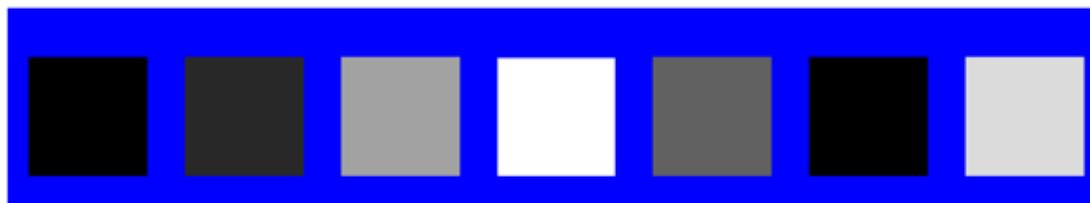
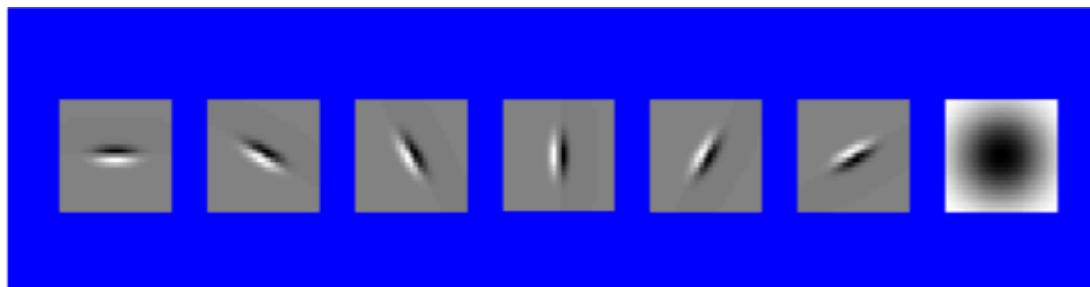


MATCH TEXTURE WITH RESPONSE



MATCH TEXTURE WITH RESPONSE

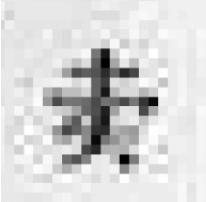
Filters



Mean abs responses

TO REMEMBER

- Image is a matrix of numbers


$$=$$


- Linear filtering is sum of dot product at each position
 - Can smooth, sharpen, translate (among many other uses)


$$\frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

- Be aware of details for filter size, extrapolation, cropping



REVIEW QUESTIONS

1. Write down a 3x3 filter that returns a positive value if the average value of the 4-adjacent neighbors is less than the center and a negative value otherwise
2. Write down a filter that will compute the gradient in the x-direction:

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

REVIEW QUESTIONS

3. Fill in the blanks:

Filtering Operator
↓

a) $\underline{\quad} = D * B$

b) $A = \underline{\quad} *$

c) $F = \overline{D} * \underline{\quad}$

d) $\underline{\quad} = D * \overline{D}$

