



GENERAL IMS LECTURE 5-6 – PIXELS AND FILTERS

Prof. Dr. Francesco Maurelli
2019-09-17
2019-09-18



WHAT WE WILL LEARN TODAY?

1. Image sampling and quantization
2. Image histograms
3. Images as functions
4. Linear systems (filters)
5. Convolution and correlation

Some background reading:
Forsyth and Ponce, Computer Vision, Chapter 7

WHAT WE WILL LEARN TODAY?

1. Image sampling and quantization
2. Image histograms
3. Images as functions
4. Linear systems (filters)
5. Convolution and correlation

Some background reading:
Forsyth and Ponce, Computer Vision, Chapter 7

Binary



Binary



Gray Scale



Binary



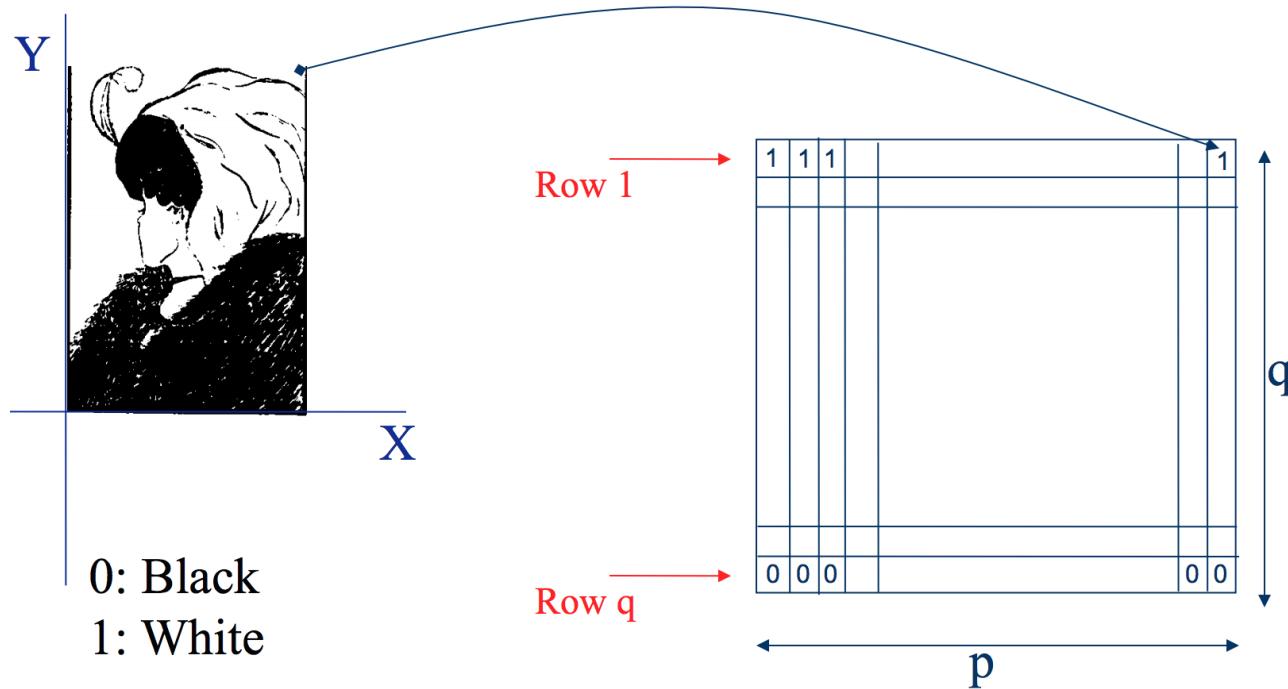
Gray Scale



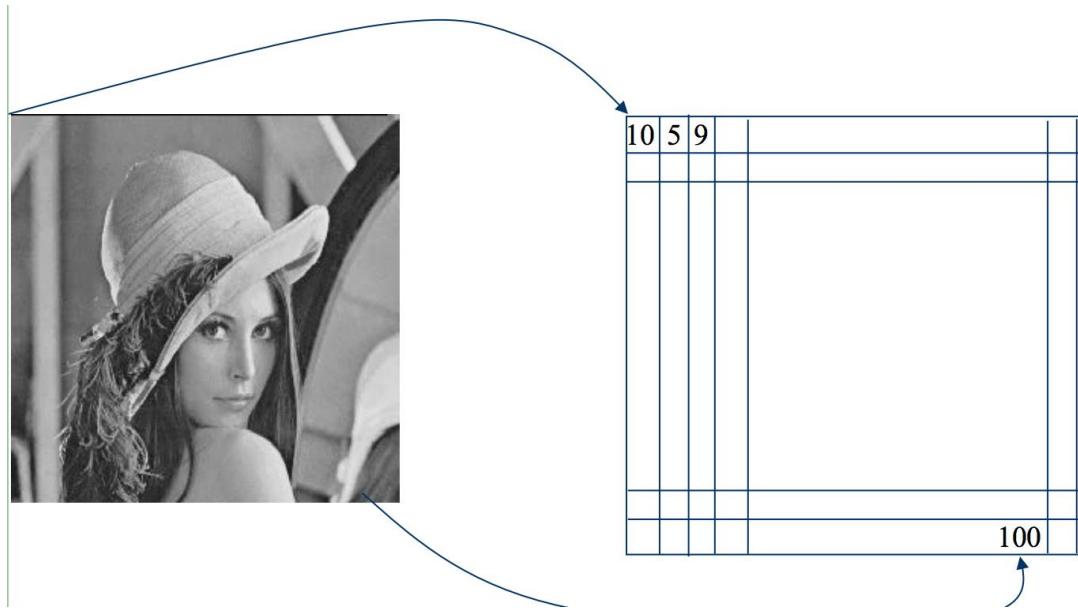
Color



BINARY IMAGE REPRESENTATION



GRAYSCALE IMAGE REPRESENTATION



COLOR IMAGE - ONE CHANNEL



COLOR IMAGE REPRESENTATION



Phil Noble / AP



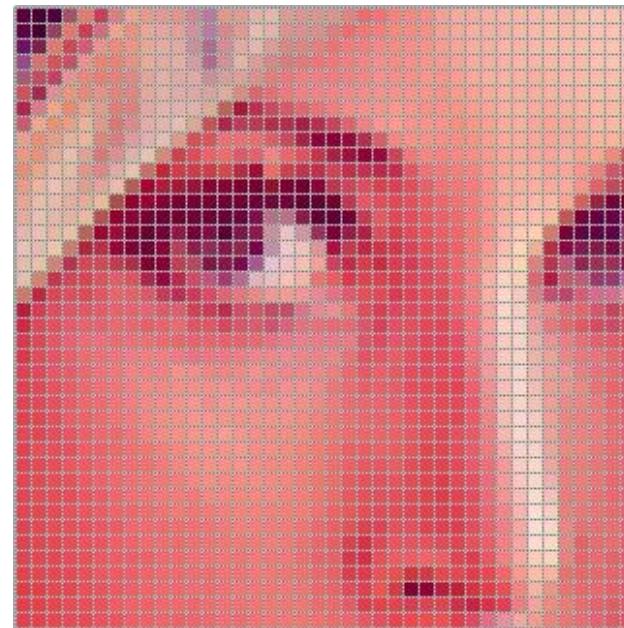
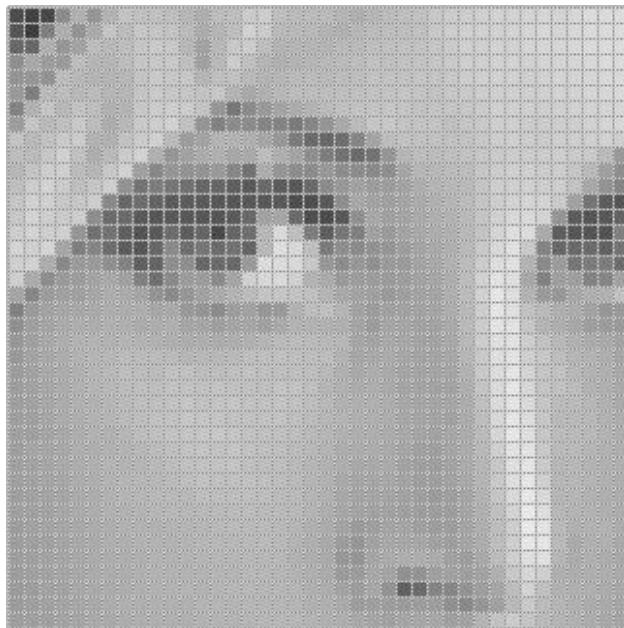
Phil Noble / AP



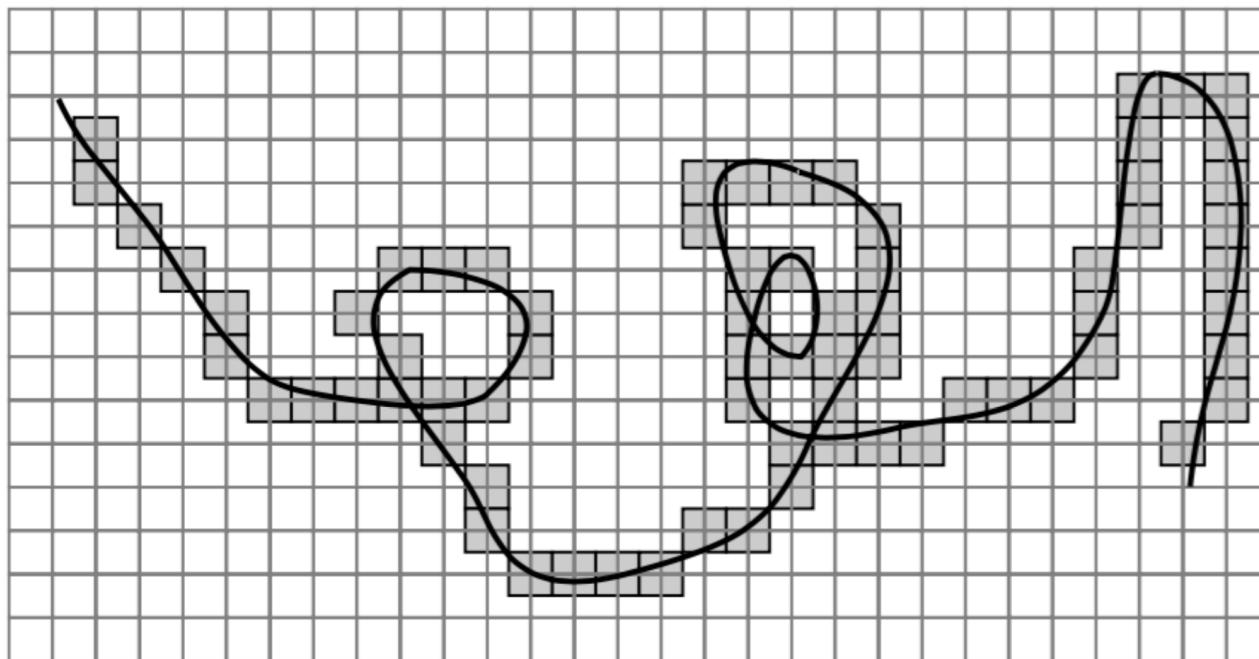
Phil Noble / AP

IMAGES ARE SAMPLED

What happens when we zoom into the images we capture?



ERRORS DUE SAMPLING



RESOLUTION

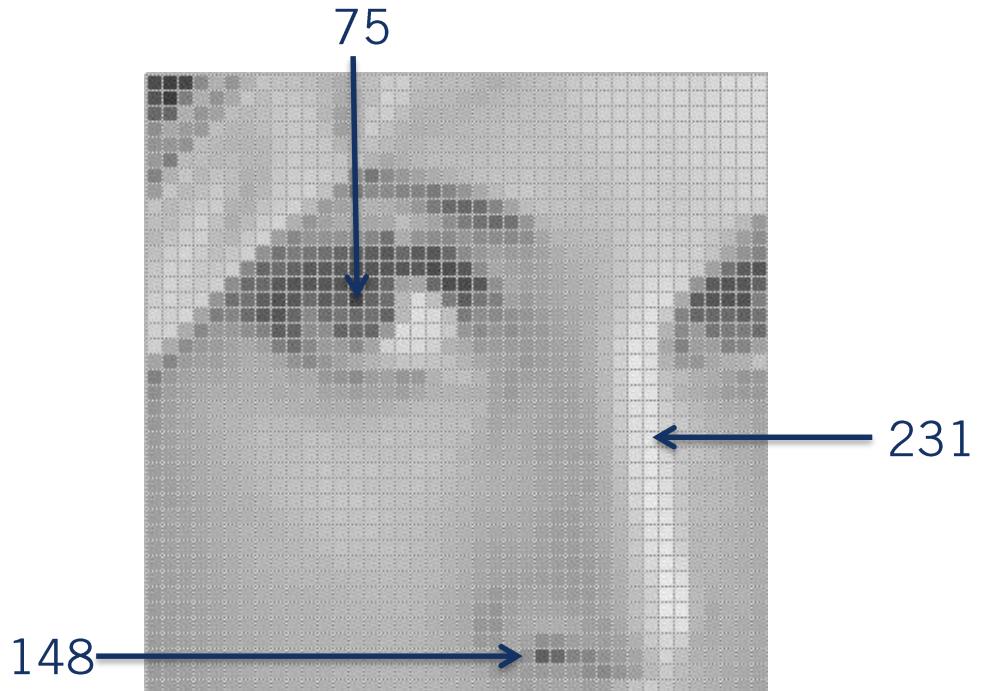
is a **sampling** parameter, defined in dots per inch (DPI) or equivalent measures of spatial pixel density, and its standard value for recent screen technologies is 72 dpi



IMAGES ARE SAMPLED AND QUANTIZED

1. An image contains discrete number of pixels

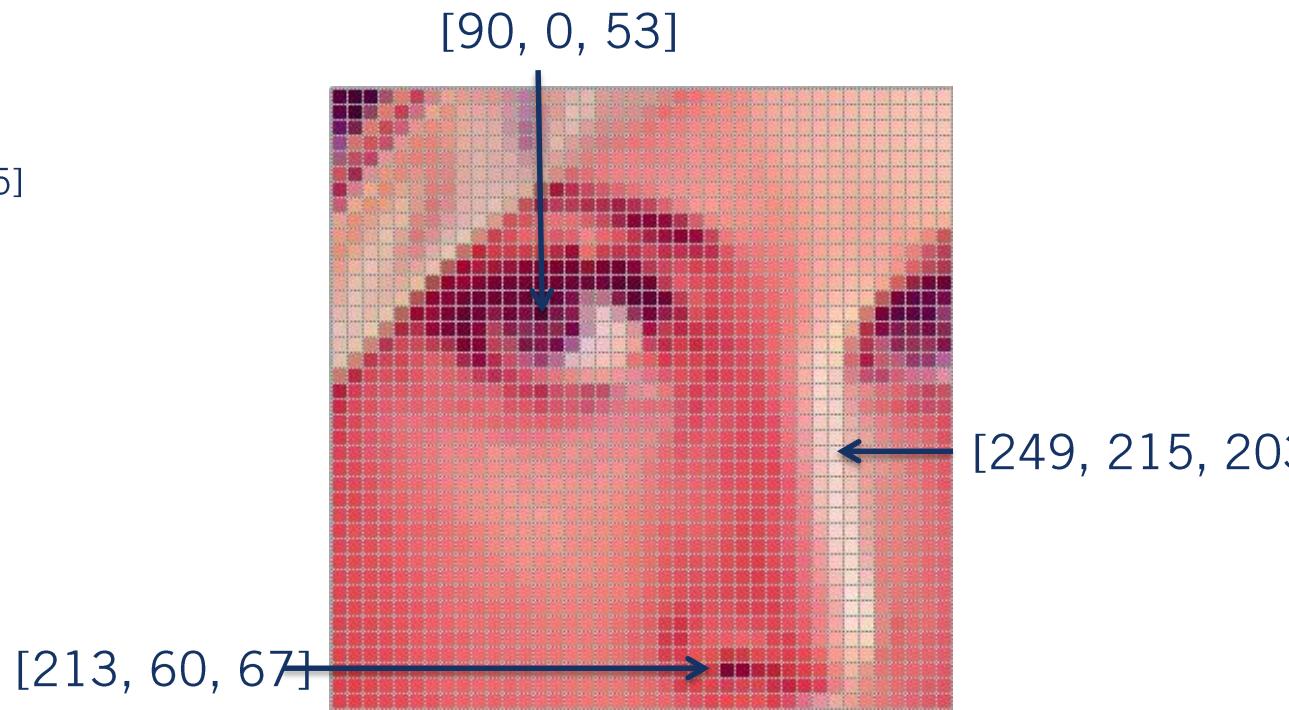
- A simple example
- Pixel value:
 - “grayscale”
(or “intensity”): [0,255]



IMAGES ARE SAMPLED AND QUANTIZED

1. An image contains discrete number of pixels

- A simple example
- Pixel value:
 - “grayscale”
(or “intensity”): [0,255]
 - “color”
 - RGB: [R, G, B]
 - Lab: [L, a, b]
 - HSV: [H, S, V]

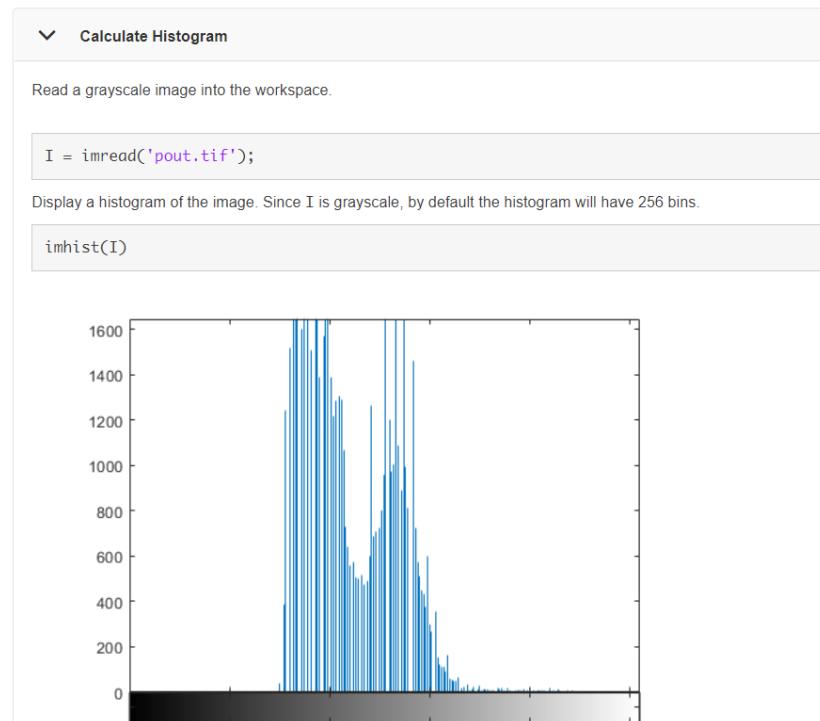


WHAT WE WILL LEARN TODAY?

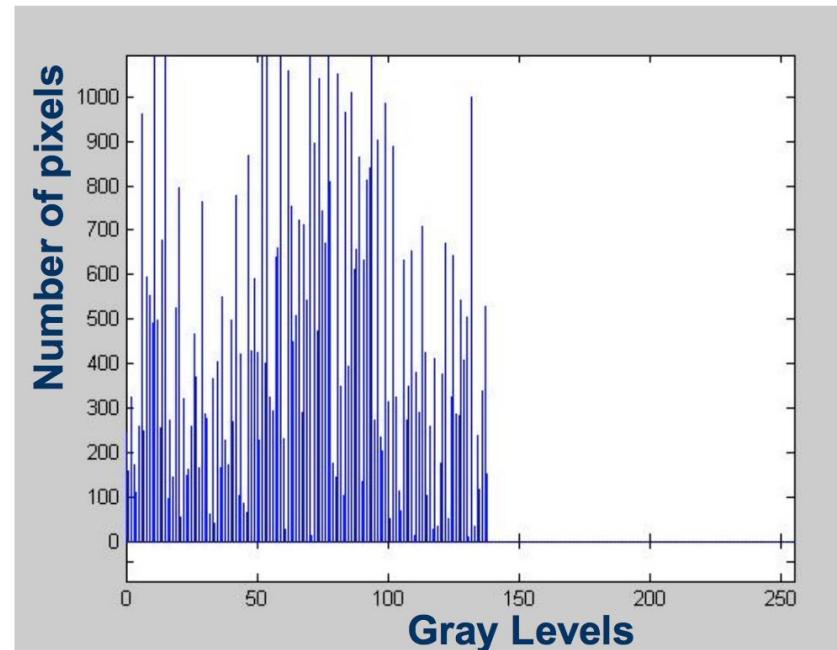
1. Image sampling and quantization
2. Image histograms
3. Images as functions
4. Linear systems (filters)
5. Convolution and correlation

Some background reading:
Forsyth and Ponce, Computer Vision, Chapter 7

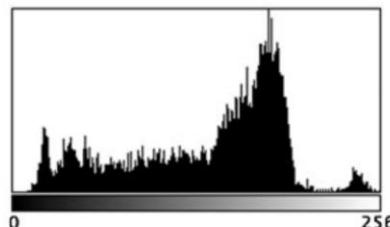
1. Histogram of an image provides the frequency of the brightness (intensity) value in the image.



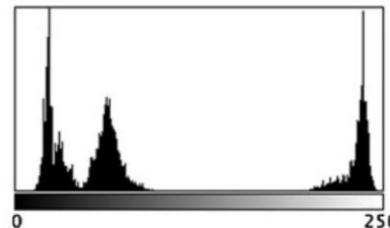
1. Histogram captures the distribution of gray levels in the image.
2. How frequently each gray level occurs in the image



HISTOGRAM

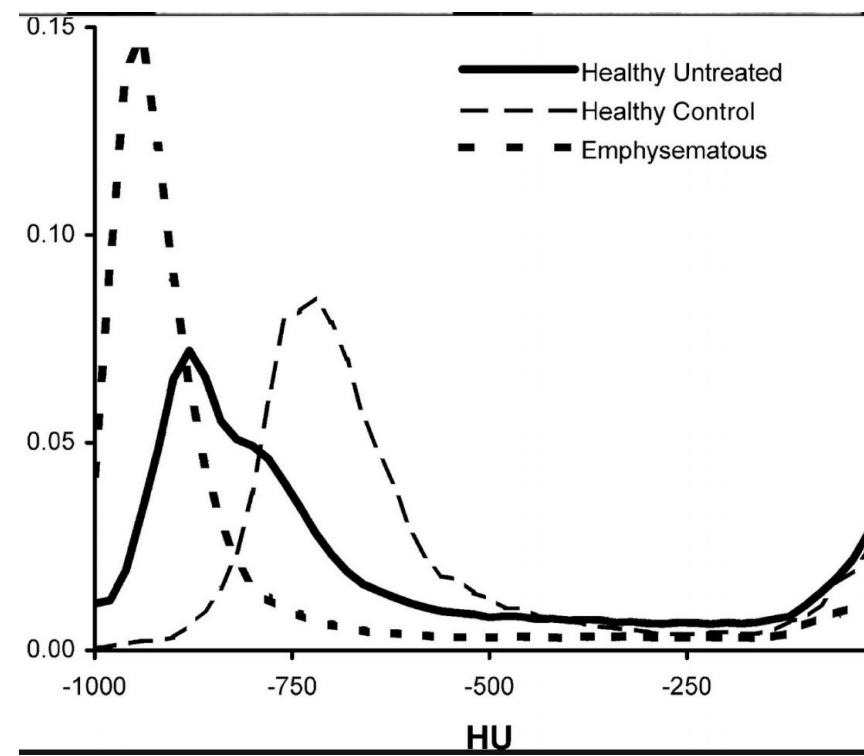
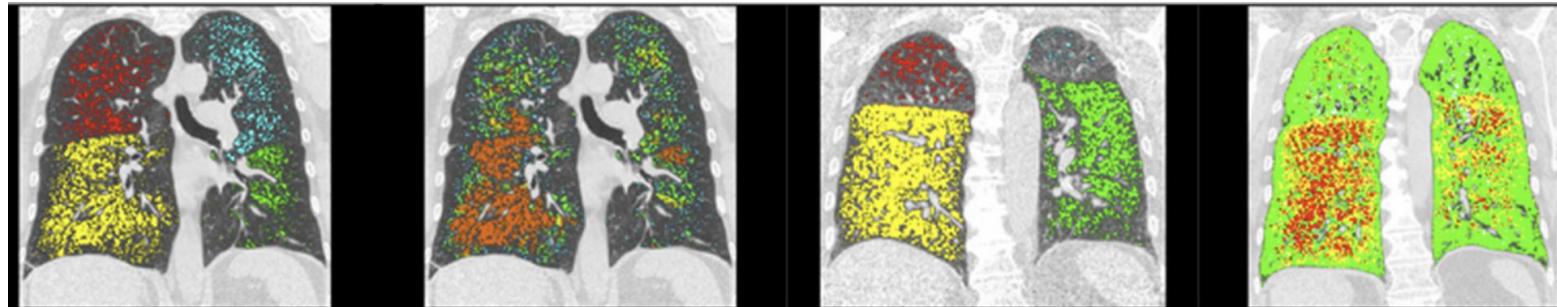


Count: 10192 Min: 9
Mean: 133.711 Max: 255
StdDev: 55.391 Mode: 178 (180)



Count: 10192 Min: 11
Mean: 104.637 Max: 254
StdDev: 89.862 Mode: 23 (440)

HISTOGRAM – USE CASE



WHAT WE WILL LEARN TODAY?

1. Image sampling and quantization
2. Image histograms
3. Images as functions
4. Linear systems (filters)
5. Convolution and correlation

Some background reading:
Forsyth and Ponce, Computer Vision, Chapter 7

- Images are usually **digital (discrete)**:
 - **Sample** the 2D space on a regular grid
- Represented as a matrix of integer values

pixel

62	79	23	119	120	05	4	0
10	10	9	62	12	78	34	0
10	58	197	46	46	0	0	48
176	135	5	188	191	68	0	49
2	1	1	29	26	37	0	77
0	89	144	147	187	102	62	208
255	252	0	166	123	62	0	31
166	63	127	17	1	0	99	30

i

j

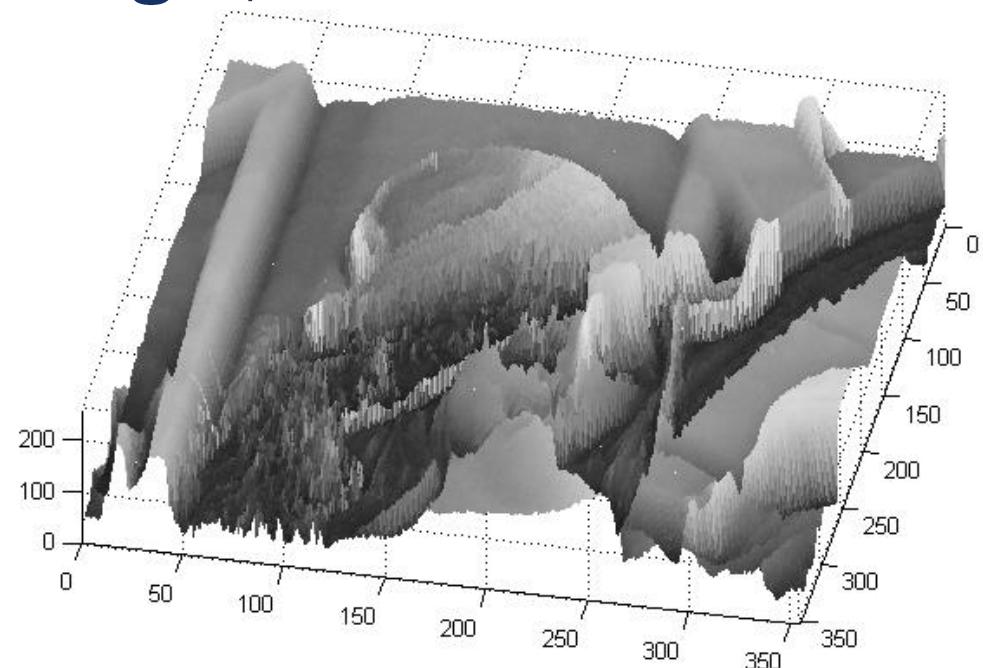
A red arrow points to the value 120 in the fifth column and second row of the matrix, which is highlighted by a red box.

Cartesian coordinates

$$f[n, m] = \begin{bmatrix} & & & & \\ \ddots & & & & \vdots \\ & f[-1, -1] & f[0, -1] & f[1, -1] & \\ \dots & f[-1, 0] & \underline{f[0, 0]} & f[1, 0] & \dots \\ & f[-1, 1] & f[0, 1] & f[1, 1] & \\ & & \vdots & & \ddots \end{bmatrix}$$

↑
Notation for discrete functions

- An Image as a function f from \mathbb{R}^2 to \mathbb{R}^M :
 - $f(x, y)$ gives the **intensity** at position (x, y)
 - Defined over a rectangle, with a finite r



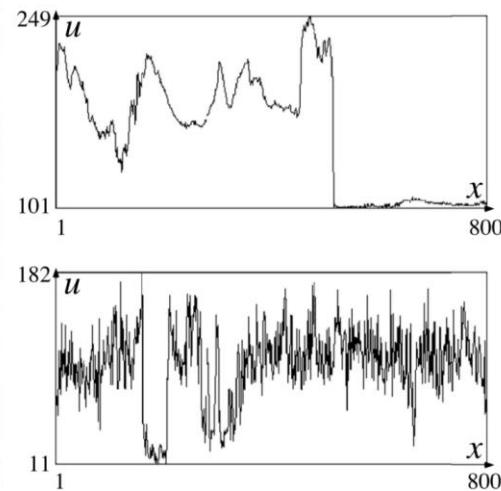
- An Image as a function f from \mathbb{R}^2 to \mathbb{R}^M :
 - $f(x, y)$ gives the **intensity** at position (x, y)
 - Defined over a rectangle, with a finite range:

$$f: [a,b] \times [c,d] \rightarrow [0,255]$$

Domain support range

- A color image: $f(x, y) = \begin{bmatrix} r(x, y) \\ g(x, y) \\ b(x, y) \end{bmatrix}$

HISTOGRAMS ARE A TYPE OF IMAGE FUNCTION



WHAT WE WILL LEARN TODAY?

1. Image sampling and quantization
2. Image histograms
3. Images as functions
4. Linear systems (filters)
5. Convolution and correlation

Some background reading:
Forsyth and Ponce, Computer Vision, Chapter 7

Filtering:

Forming a new image whose pixel values are transformed from original pixel values

Goals:

To extract useful information from images, or transform images into another domain where we can modify/enhance image properties

- Features (edges, corners, blobs...)
- super-resolution; in-painting; de-noising

we define a system as a unit that converts an input function $f[n,m]$ into an output (or response) function $g[n,m]$, where (n,m) are the independent variables.

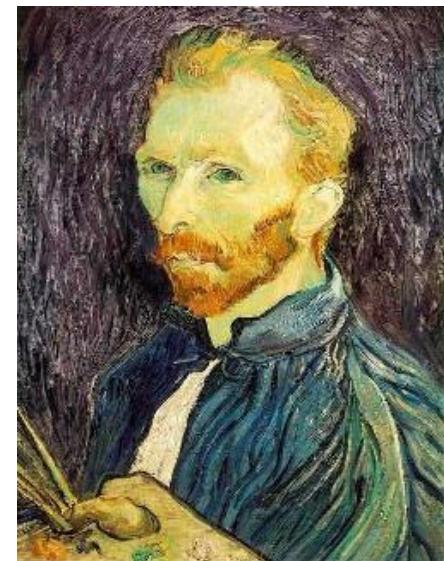
In the case for images, (n,m) represents the **spatial position in the image**.

$$f[n, m] \rightarrow \boxed{\text{System } \mathcal{S}} \rightarrow g[n, m]$$

DE-NOISING



Salt and pepper noise



Super-resolution



Image Inpainting, M. Bertalmio et al.
<http://www.ima.upf.es/~mbertalmio/restoration.html>

In-painting

S is the **system operator**, defined as a mapping or assignment of a member of the set of possible outputs $g[n,m]$ to each member of the set of possible inputs $f[n,m]$.

$$f[n, m] \rightarrow \boxed{\text{System } S} \rightarrow g[n, m]$$

$$g = \mathcal{S}[f], \quad g[n, m] = \mathcal{S}\{f[n, m]\}$$

$$f[n, m] \xrightarrow{S} g[n, m]$$

FILTER EXAMPLE #1: MOVING AVERAGE

Moving average over a 3×3 window of neighborhood

$$g[n, m] = \frac{1}{9} \sum_{k=n-1}^{n+1} \sum_{l=m-1}^{m+1} f[k, l]$$

$$= \frac{1}{9} \sum_{k=-1}^1 \sum_{l=-1}^1 f[n - k, m - l]$$

$$\frac{1}{9} \begin{matrix} h \\ \hline \begin{matrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{matrix} \end{matrix}$$

FILTER EXAMPLE #1: MOVING AVERAGE

$F[x, y]$

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

$G[x, y]$

$$(f^* h)[m, n] = \sum_{k,l} f[k, l] h[m - k, n - l]$$

FILTER EXAMPLE #1: MOVING AVERAGE

$F[x, y]$

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

$G[x, y]$

0	10								

$$(f^* h)[m, n] = \sum_{k,l} f[k, l] h[m - k, n - l]$$

FILTER EXAMPLE #1: MOVING AVERAGE

$F[x, y]$

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	90	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

$G[x, y]$

0	10	20							

$$(f^* h)[m, n] = \sum_{k,l} f[k, l] h[m - k, n - l]$$

FILTER EXAMPLE #1: MOVING AVERAGE

 $F[x, y]$

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

 $G[x, y]$

$$(f^* h)[m, n] = \sum_{k,l} f[k, l] h[m - k, n - l]$$

 k, l

$$F[x, y]$$

$$G[x, y]$$

$$(f^* h)[m, n] = \sum_{k,l} \hat{a}_f[k, l] h[m - k, n - l]$$

FILTER EXAMPLE #1: MOVING AVERAGE

$F[x, y]$

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

$G[x, y]$

	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		

$$(f^* h)[m, n] = \sum_{k,l} f[k, l] h[m - k, n - l]$$

k, l

In summary:

- This filter “Replaces” each pixel with an average of its neighborhood.
- Achieve smoothing effect (remove sharp features)

$$h[\times, \times] = \frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

FILTER EXAMPLE #1: MOVING AVERAGE



FILTER EXAMPLE #2: IMAGE SEGMENTATION

Image segmentation based on a simple threshold:

$$g[n, m] = \begin{cases} 255, & f[n, m] > 100 \\ 0, & \text{otherwise.} \end{cases}$$



Amplitude properties:

- Additivity

$$S[f_i[n, m] + f_j[n, m]] = S[f_i[n, m]] + S[f_j[n, m]]$$

- Homogeneity

$$S[\alpha f_i[n, m]] = \alpha S[f_i[n, m]]$$

- Superposition

$$S[\alpha f_i[n, m] + \beta f_j[n, m]] = \alpha S[f_i[n, m]] + \beta S[f_j[n, m]]$$

- Stability

$$|f[n, m]| \leq k \implies |g[n, m]| \leq ck$$

- Invertibility

$$S^{-1}[S[f_i[n, m]]] = f[n, m]$$

Spatial properties

- Causality for $n < n_0, m < m_0$, if $f[n, m] = 0 \implies g[n, m] = 0$
- Shift invariance

$$f[n - n_0, m - m_0] \xrightarrow{\mathcal{S}} g[n - n_0, m - m_0]$$

IS THE MOVING AVERAGE SYSTEM SHIFT INVARIANT?

$$f[n, m] \xrightarrow{\mathcal{S}} g[n, m] = \frac{1}{9} \sum_{k=-1}^1 \sum_{l=-1}^1 f[n - k, m - l]$$

$F[x, y]$

$G[x, y]$

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

	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		

IS THE MOVING AVERAGE SYSTEM SHIFT INVARIANT?

$$f[n, m] \xrightarrow{\mathcal{S}} g[n, m] = \frac{1}{9} \sum_{k=-1}^1 \sum_{l=-1}^1 f[n - k, m - l]$$

$$f[n - n_0, m - m_0]$$

$$\xrightarrow{\mathcal{S}} \frac{1}{9} \sum_{k=-1}^1 \sum_{l=-1}^1 f[(n - n_0) - k, (m - m_0) - l]$$

$$= g[n - n_0, m - m_0]$$

Yes!

IS THE MOVING AVERAGE SYSTEM IS CASUAL?

$$f[n, m] \xrightarrow{\mathcal{S}} g[n, m] = \frac{1}{9} \sum_{k=-1}^1 \sum_{l=-1}^1 f[n - k, m - l]$$

$F[x, y]$

$G[x, y]$

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

			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

for $n < n_0, m < m_0$, if $f[n, m] = 0 \implies g[n, m] = 0$

$$f[n, m] \rightarrow \boxed{\text{System } S} \rightarrow g[n, m]$$

- Linear filtering:
 - Form a new image whose pixels are a weighted sum of original pixel values
 - Use the same set of weights at each point
- **S** is a linear system (function) iff it *S* satisfies

$$S[\alpha f_i[n, m] + \beta f_j[h, m]] = \alpha S[f_i[n, m]] + \beta S[f_j[h, m]]$$

superposition property

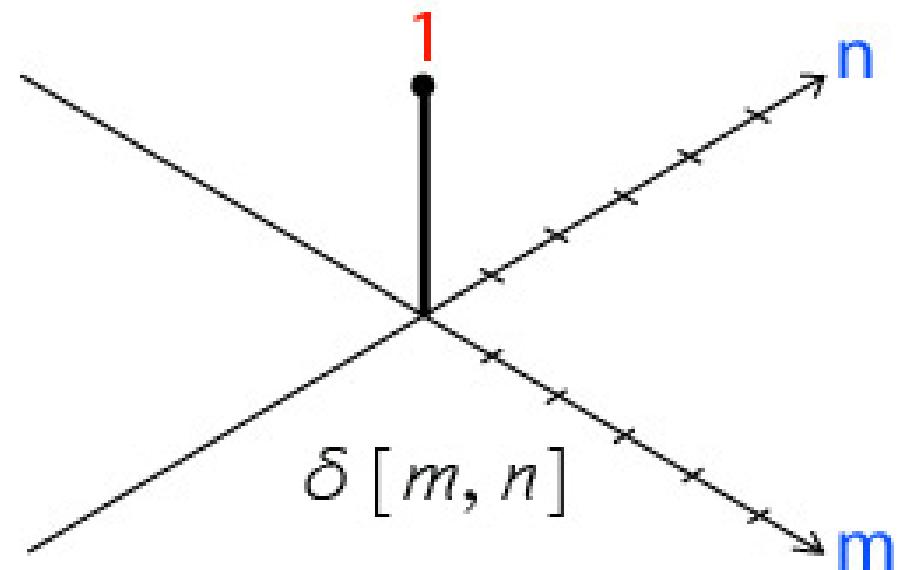
$$f[n, m] \rightarrow \boxed{\text{System } \mathcal{S}} \rightarrow g[n, m]$$

- Is the moving average a linear system?
- Is thresholding a linear system?
 - $f_1[n, m] + f_2[n, m] > T$
 - $f_1[n, m] < T$
 - $f_2[n, m] < T$

No!

2D IMPULSE FUNCTION

1. 1 at [0,0].
2. 0 everywhere else



Impulse response

$$\delta_2[n, m] \rightarrow \boxed{\mathcal{S}} \rightarrow h[n, m]$$

$$\delta_2[n - k, m - l] \rightarrow \boxed{\mathcal{S} \text{ (SI)}} \rightarrow h[n - k, m - l]$$

Example: impulse response of the 3 by 3 moving average filter:

$$h[n, m] = \frac{1}{9} \sum_{k=-1}^1 \sum_{l=-1}^1 \delta_2[n - k, m - l]$$
$$= \begin{bmatrix} 1/9 & 1/9 & 1/9 \\ 1/9 & 1/9 & 1/9 \\ 1/9 & 1/9 & 1/9 \end{bmatrix}$$

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

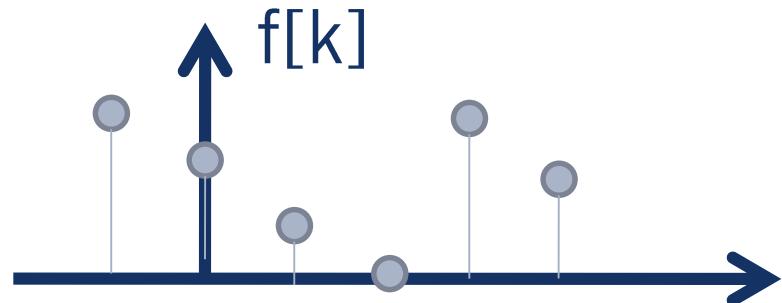
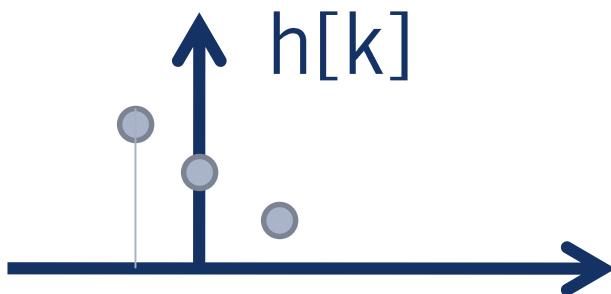
WHAT WE WILL LEARN TODAY?

1. Images as functions
2. Linear systems (filters)
3. Convolution and correlation

Some background reading:
Forsyth and Ponce, Computer Vision, Chapter 7

We are going to convolve a function \mathbf{f} with a filter \mathbf{h} .

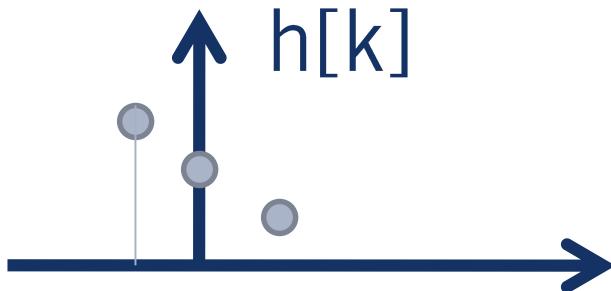
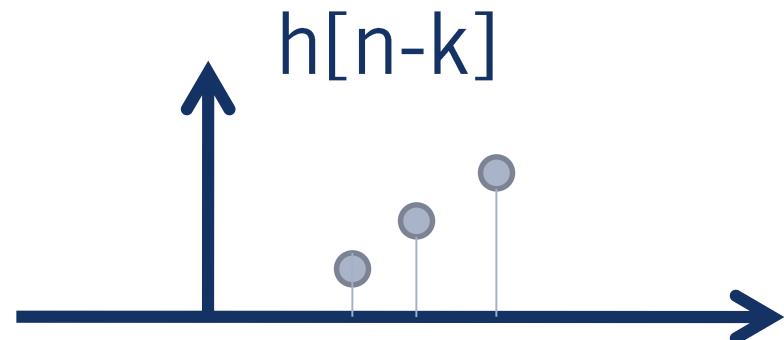
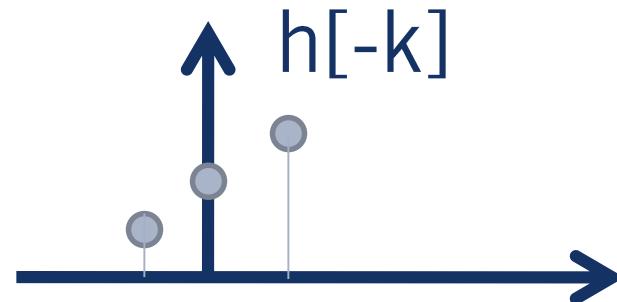
$$g[n] = \sum_k f[k]h[n - k]$$



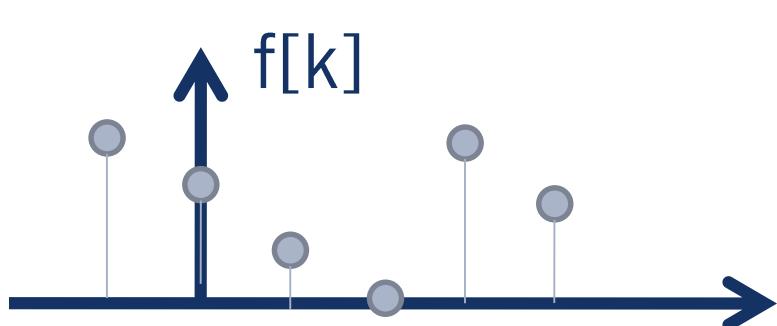
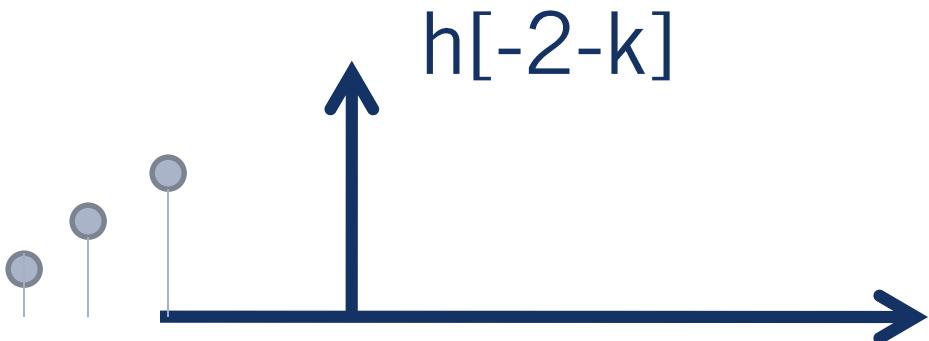
We are going to convolve a function f with a filter h .

$$g[n] = \sum_k f[k]h[n - k]$$

We first need to calculate $h[n-k]$

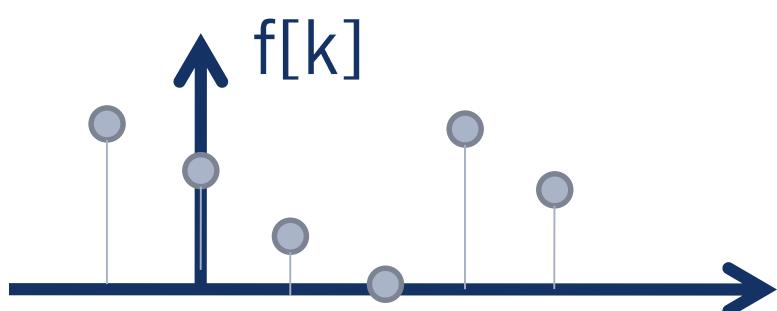
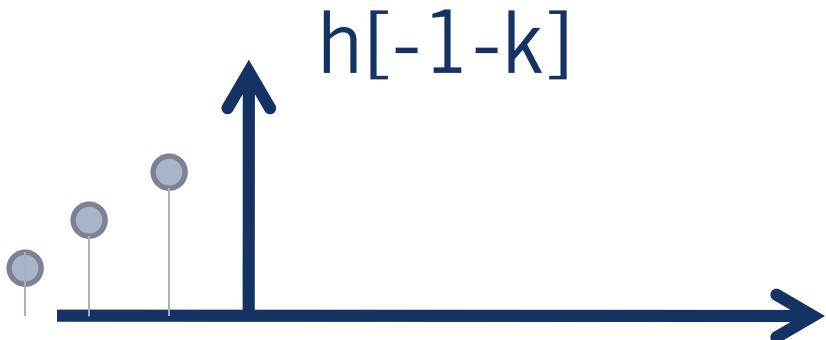


We are going to convolve a function f with a filter h .



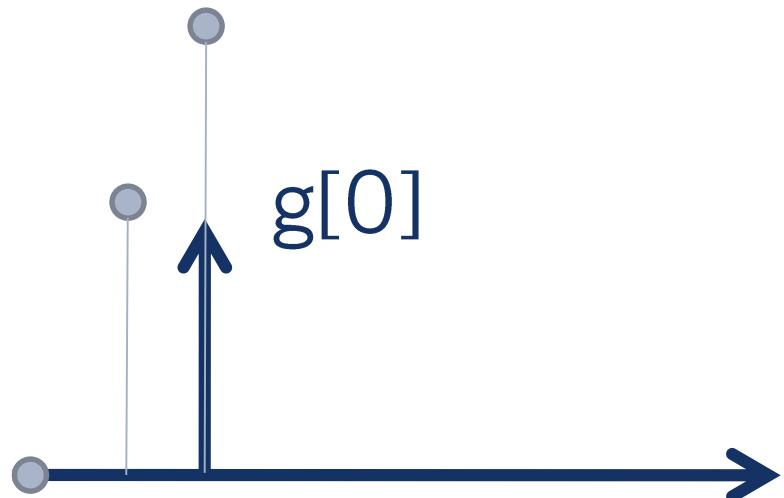
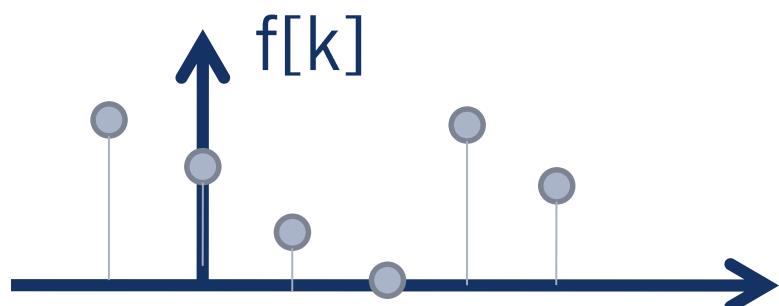
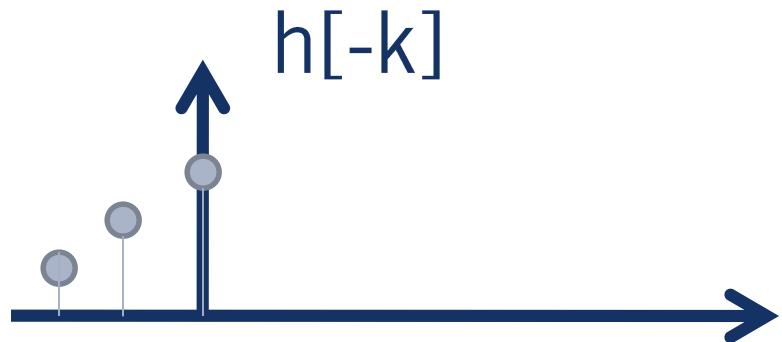
1D DISCRETE CONVOLUTION (SYMBOL: *)

We are going to convolve a function f with a filter h .



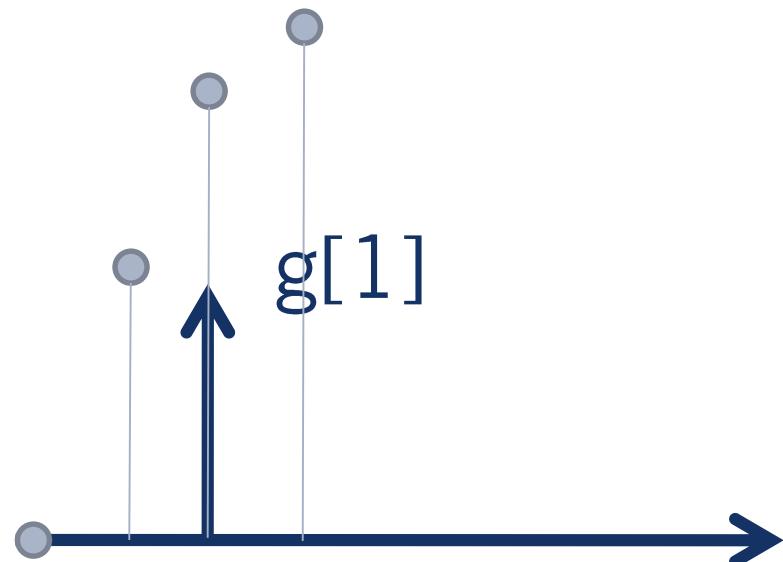
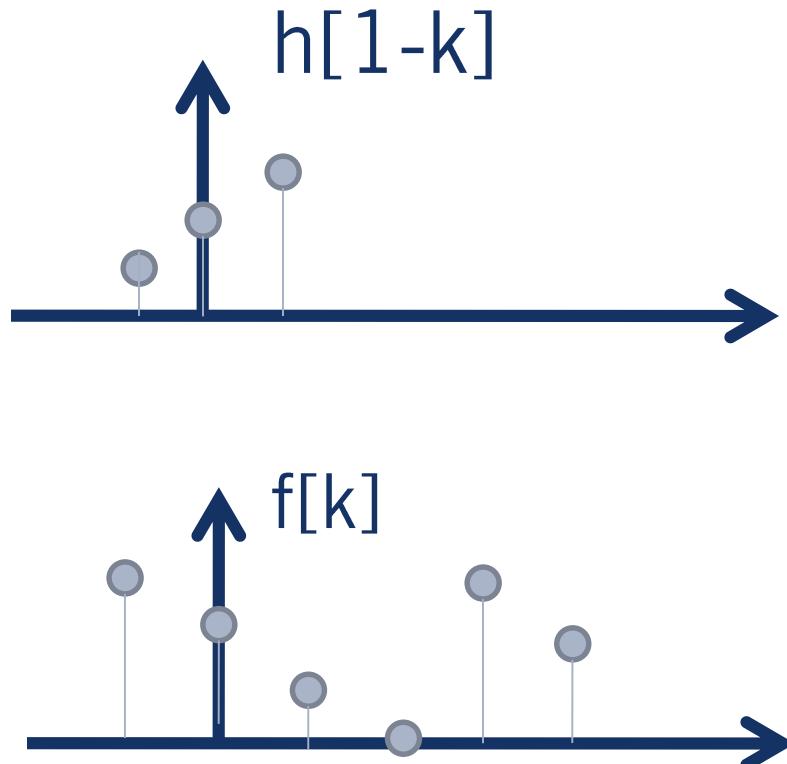
1D DISCRETE CONVOLUTION (SYMBOL: *)

We are going to convolve a function f with a filter h .



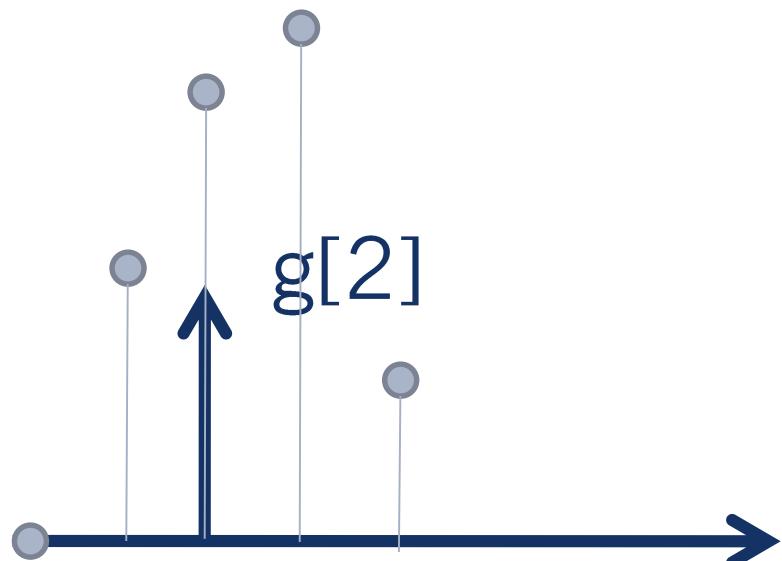
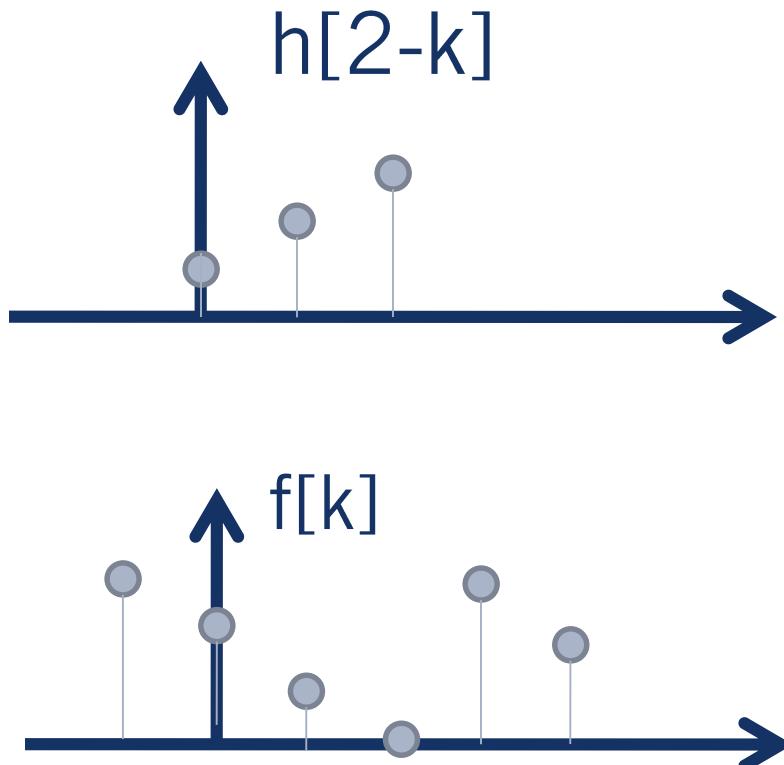
1D DISCRETE CONVOLUTION (SYMBOL: *)

We are going to convolve a function f with a filter h .

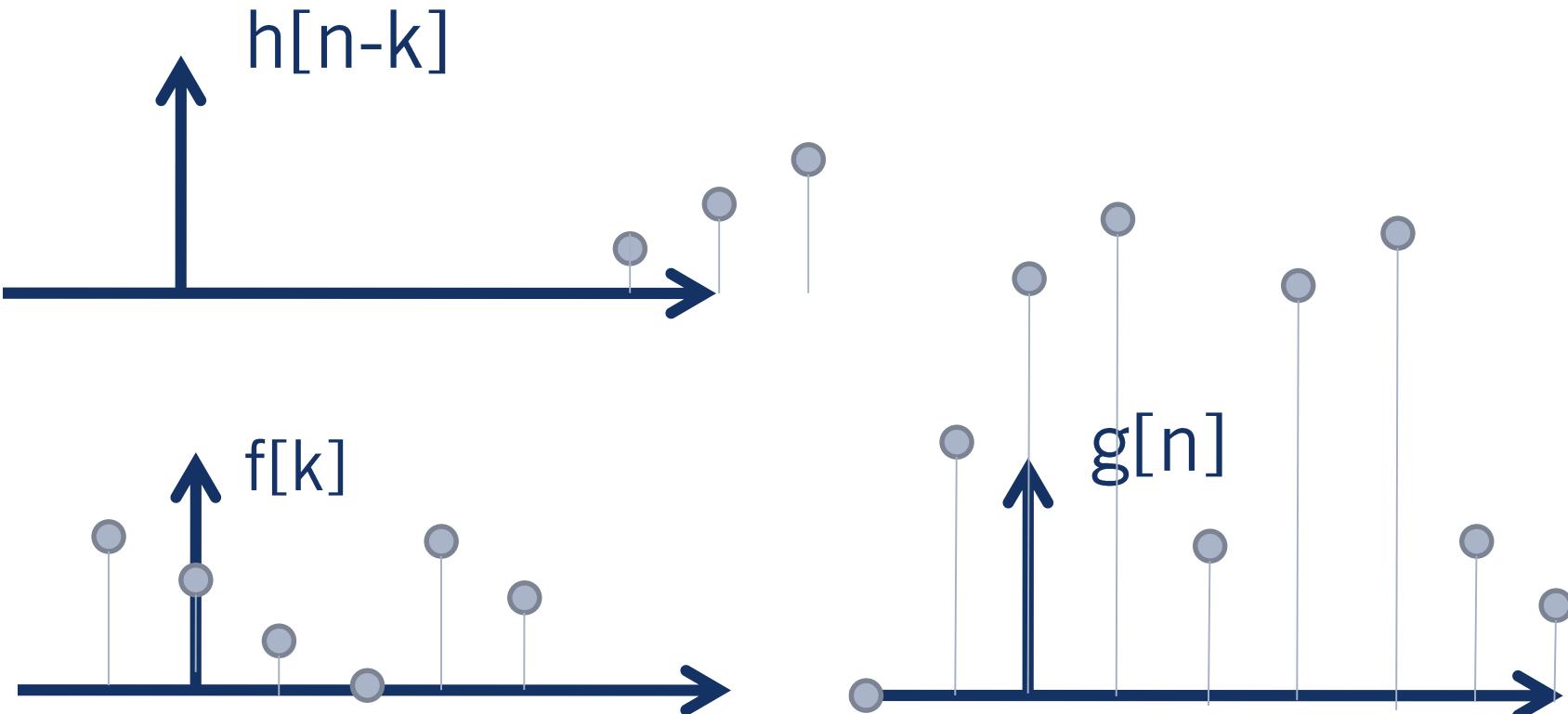


1D DISCRETE CONVOLUTION (SYMBOL: *)

We are going to convolve a function f with a filter h .



We are going to convolve a function f with a filter h .



1D DISCRETE CONVOLUTION (SYMBOL: *)

In summary, the steps for discrete convolution are:

- Fold $h[k]$ about origin to form $h[-k]$
- Shift the folded results by n to form $h[n - k]$
- Multiply $h[n - k]$ by $f[k]$
- Sum over all k
- Repeat for every n

$$g[n] = \sum_k f[k]h[n - k]$$

1D DISCRETE CONVOLUTION (SYMBOL: *)

In a different notation (g is the kernel now)

$$f = \boxed{10 \ 50 \ 60 \ 10 \ 20 \ 40 \ 30}$$

$$g = \boxed{1/3 \ 1/3 \ 1/3}$$

$$\begin{array}{|c|c|c|c|c|c|c|} \hline 10 & 50 & 60 & 10 & 20 & 40 & 30 \\ \hline 1/3 & 1/3 & 1/3 & & & & \\ \hline \end{array}$$

$$\begin{array}{|c|c|c|c|c|c|c|} \hline 10 & 50 & 60 & 10 & 20 & 30 & 40 \\ \hline 0 & 1/3 & 1/3 & 1/3 & 0 & 0 & 0 \\ \hline \end{array}$$

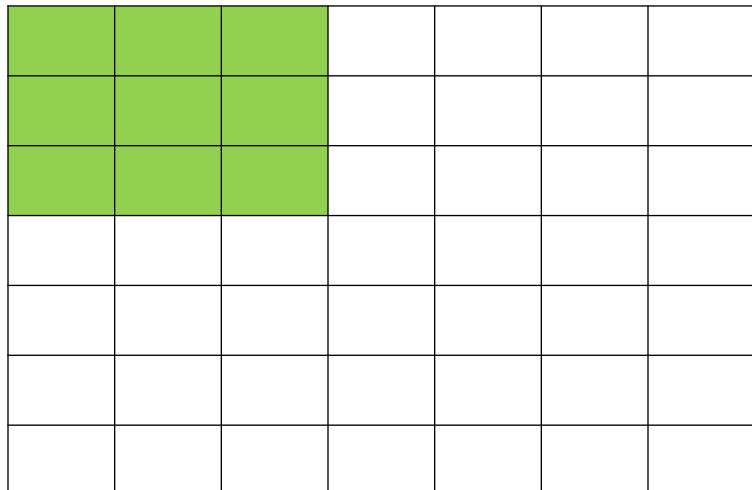
$$\frac{1}{3}50 + \frac{1}{3}60 + \frac{1}{3}10 = \frac{50}{3} + \frac{60}{3} + \frac{10}{3} = \frac{120}{3} = 40$$

n

2D convolution is very similar to 1D.

The main difference is that we now have to iterate over 2 axis instead of 1.

$$f[n, m] * h[n, m] = \sum_{k=-\infty}^{\infty} \sum_{l=-\infty}^{\infty} f[k, l] h[n - k, m - l]$$

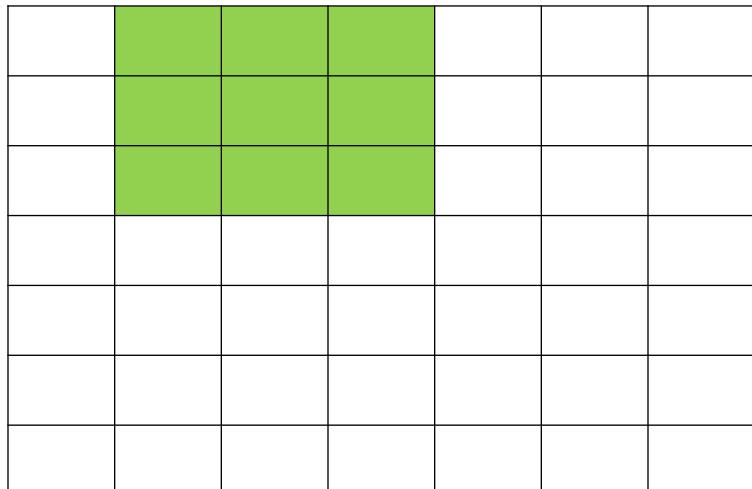


Assume we have a filter($h[,]$)
that is 3x3. and an image ($f[,]$)
that is 7x7.

2D convolution is very similar to 1D.

- The main difference is that we now have to iterate over 2 axis instead of 1.

$$f[n, m] * h[n, m] = \sum_{k=-\infty}^{\infty} \sum_{l=-\infty}^{\infty} f[k, l] h[n - k, m - l]$$

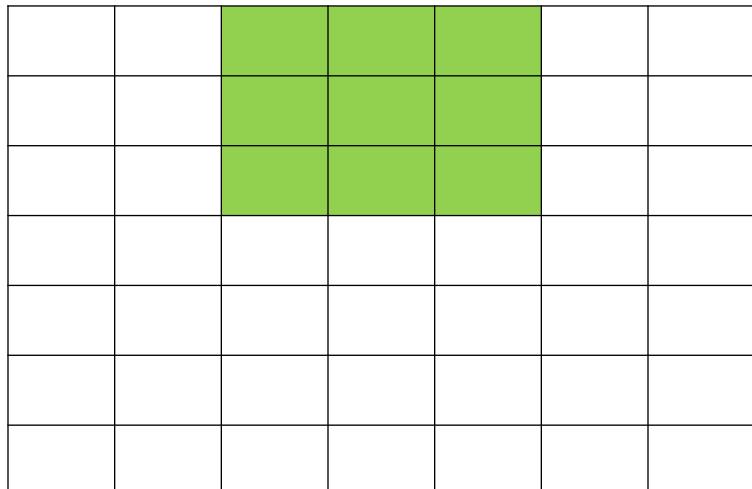


Assume we have a filter($h[,]$)
that is 3x3. and an image ($f[,]$)
that is 7x7.

2D convolution is very similar to 1D.

- The main difference is that we now have to iterate over 2 axis instead of 1.

$$f[n, m] * h[n, m] = \sum_{k=-\infty}^{\infty} \sum_{l=-\infty}^{\infty} f[k, l] h[n - k, m - l]$$

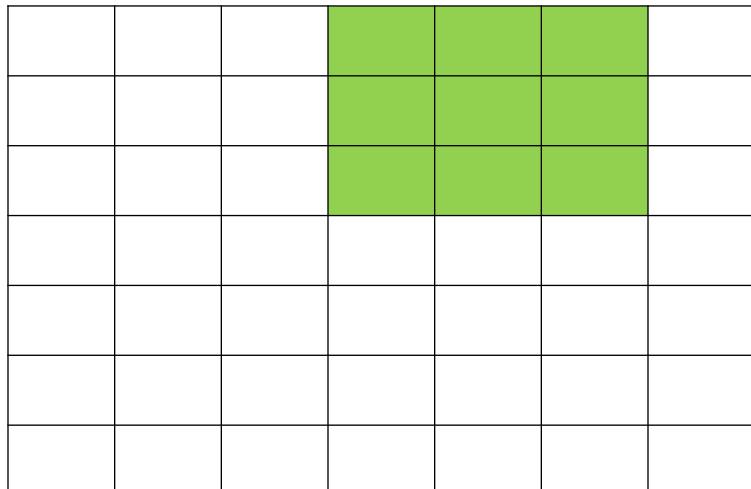


Assume we have a filter($h[,]$)
that is 3x3. and an image ($f[,]$)
that is 7x7.

2D convolution is very similar to 1D.

- The main difference is that we now have to iterate over 2 axis instead of 1.

$$f[n, m] * h[n, m] = \sum_{k=-\infty}^{\infty} \sum_{l=-\infty}^{\infty} f[k, l] h[n - k, m - l]$$

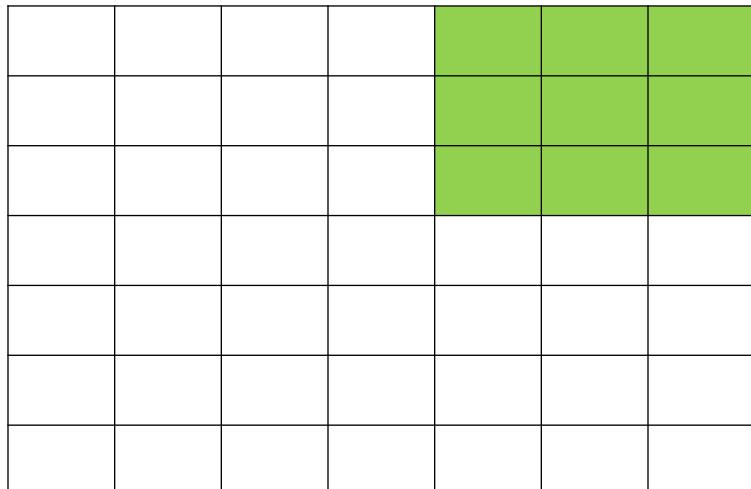


Assume we have a filter($h[,]$) that is 3x3. and an image ($f[,]$) that is 7x7.

2D convolution is very similar to 1D.

- The main difference is that we now have to iterate over 2 axis instead of 1.

$$f[n, m] * h[n, m] = \sum_{k=-\infty}^{\infty} \sum_{l=-\infty}^{\infty} f[k, l] h[n - k, m - l]$$

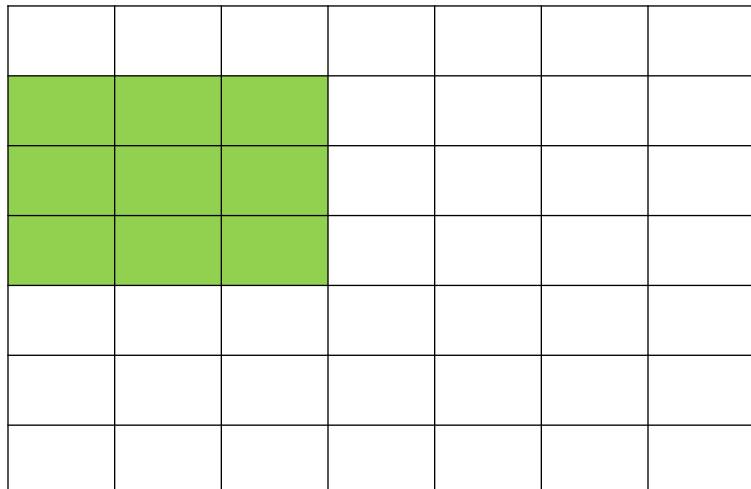


Assume we have a filter($h[,]$) that is 3x3. and an image ($f[,]$) that is 7x7.

2D convolution is very similar to 1D.

- The main difference is that we now have to iterate over 2 axis instead of 1.

$$f[n, m] * h[n, m] = \sum_{k=-\infty}^{\infty} \sum_{l=-\infty}^{\infty} f[k, l] h[n - k, m - l]$$



Assume we have a filter($h[,]$) that is 3x3. and an image ($f[,]$) that is 7x7.

An LSI system is completely specified by its impulse response.

shifting property of the delta function

$$f[n, m] = \sum_{k=-\infty}^{\infty} \sum_{l=-\infty}^{\infty} f[k, l] \delta_2[n - k, m - l]$$

superposition

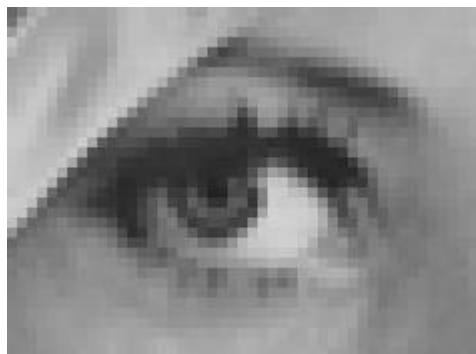
$$\rightarrow \boxed{\mathcal{S} \text{ LSI}} \rightarrow \sum_{k=-\infty}^{\infty} \sum_{l=-\infty}^{\infty} f[k, l] h[n - k, m - l]$$

$\delta_2[n, m] \rightarrow \boxed{\mathcal{S}} \rightarrow h[n, m]$

Discrete convolution

$$f[n, m] * h[n, m]$$

CONVOLUTION IN 2D - EXAMPLES



*

$$\begin{array}{|c|c|c|} \hline \bullet & 0 & 0 \\ \hline 0 & \bullet & 1 \\ \hline 0 & 0 & 0 \\ \hline \end{array}$$

=

?

Original

CONVOLUTION IN 2D - EXAMPLES



*

$$\begin{array}{|c|c|c|} \hline \bullet & 0 & 0 \\ \hline 0 & \bullet & 1 \\ \hline 0 & 0 & 0 \\ \hline \end{array}$$

Original

=



Filtered
(no change)

CONVOLUTION IN 2D - EXAMPLES



*

•0	•0	•0
•0	•0	•1
•0	•0	•0

=

?

Original

CONVOLUTION IN 2D - EXAMPLES



*

•0	•0	•0
•0	•0	•1
•0	•0	•0

=



Original

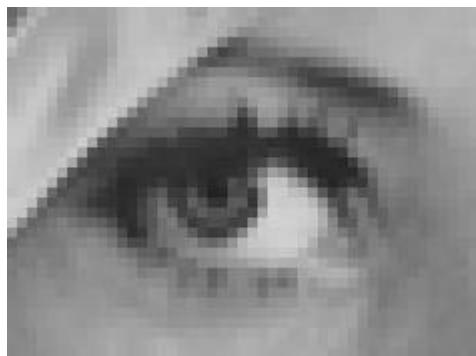
Shifted right
By 1 pixel

CONVOLUTION IN 2D - EXAMPLES



$$\text{Original} \quad * \frac{1}{9} \begin{matrix} \bullet 1 & \bullet 1 & \bullet 1 \\ \bullet 1 & \bullet 1 & \bullet 1 \\ \bullet 1 & \bullet 1 & \bullet 1 \end{matrix} = ?$$

CONVOLUTION IN 2D - EXAMPLES



Original

$$\ast \frac{1}{9} \begin{matrix} \bullet 1 & \bullet 1 & \bullet 1 \\ \bullet 1 & \bullet 1 & \bullet 1 \\ \bullet 1 & \bullet 1 & \bullet 1 \end{matrix} =$$



Blur (with a
box filter)

CONVOLUTION IN 2D - EXAMPLES



Original

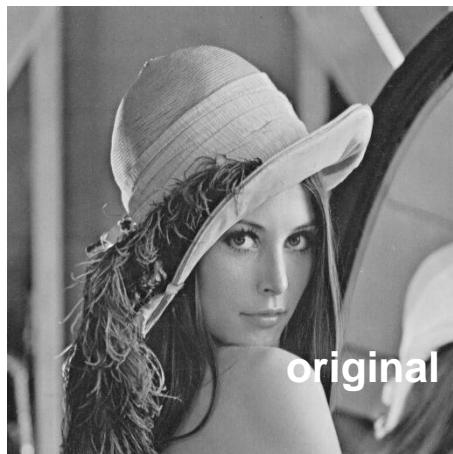
$$\begin{matrix}
 \bullet & 0 & \bullet & 0 & \bullet & 0 \\
 \bullet & 0 & \bullet & 2 & \bullet & 0 \\
 \bullet & 0 & \bullet & 0 & \bullet & 0
 \end{matrix} \quad - \quad \frac{1}{9} \begin{matrix}
 \bullet & 1 & \bullet & 1 & \bullet & 1 \\
 \bullet & 1 & \bullet & 1 & \bullet & 1 \\
 \bullet & 1 & \bullet & 1 & \bullet & 1
 \end{matrix} = ?$$

(Note that filter sums to 1)

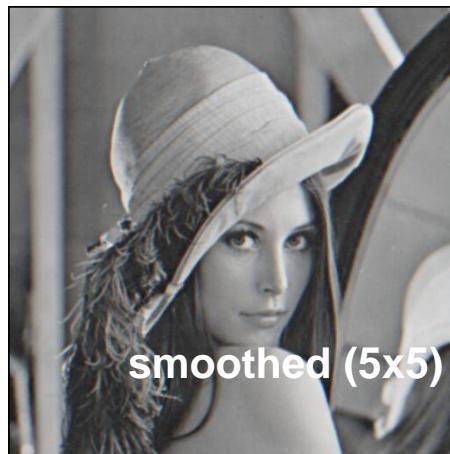
“details of the image”

$$\begin{matrix}
 \bullet & 0 & \bullet & 0 & \bullet & 0 \\
 \bullet & 0 & \bullet & 1 & \bullet & 0 \\
 \bullet & 0 & \bullet & 0 & \bullet & 0
 \end{matrix} + \begin{matrix}
 \bullet & 0 & \bullet & 0 & \bullet & 0 \\
 \bullet & 0 & \bullet & 1 & \bullet & 0 \\
 \bullet & 0 & \bullet & 0 & \bullet & 0
 \end{matrix} - \frac{1}{9} \begin{matrix}
 \bullet & 1 & \bullet & 1 & \bullet & 1 \\
 \bullet & 1 & \bullet & 1 & \bullet & 1 \\
 \bullet & 1 & \bullet & 1 & \bullet & 1
 \end{matrix}$$

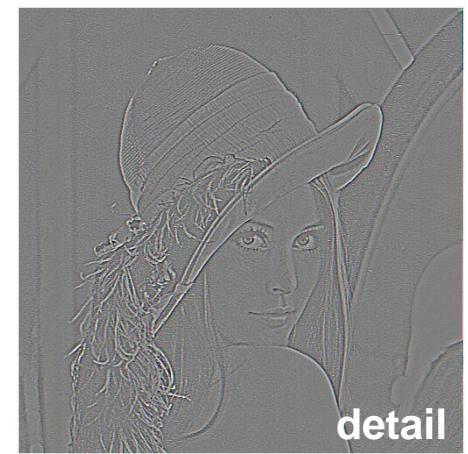
WHAT DOES BLURRING TAKE AWAY?



-



=



- Let's add it back:



+ a



=



CONVOLUTION IN 2D – SHARPENING FILTER



Original

$$\begin{bmatrix} \bullet 0 & \bullet 0 & \bullet 0 \\ \bullet 0 & \bullet 2 & \bullet 0 \\ \bullet 0 & \bullet 0 & \bullet 0 \end{bmatrix}$$

-

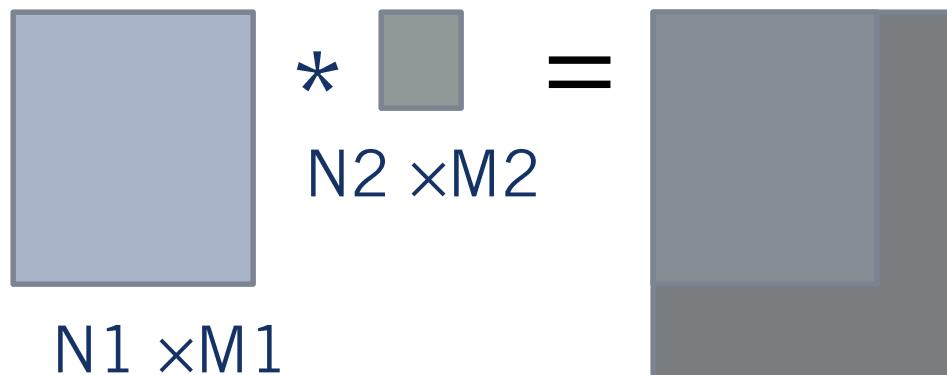
$$\frac{1}{9} \begin{bmatrix} \bullet 1 & \bullet 1 & \bullet 1 \\ \bullet 1 & \bullet 1 & \bullet 1 \\ \bullet 1 & \bullet 1 & \bullet 1 \end{bmatrix}$$

=



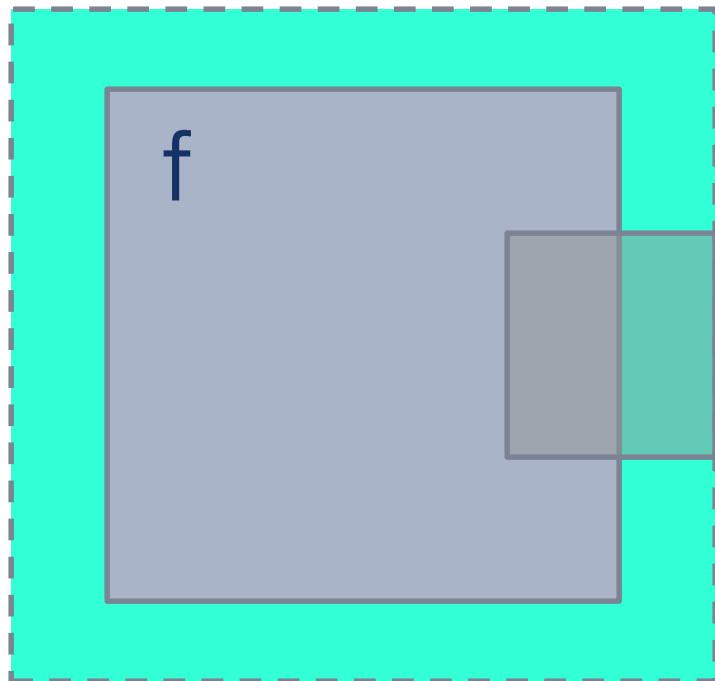
Sharpening filter: Accentuates differences with local average

- A computer will only convolve **finite support signals.**
 - That is: images that are zero for n, m outside some rectangular region
 - MATLAB's conv2 performs 2D DS convolution of finite-support signals.



$$(N_1 + N_2 - 1) \times (M_1 + M_2 - 1)$$

- A computer will only convolve **finite support signals.**
- What happens at the edge?



- zero “padding”
 - edge value replication
 - mirror extension
 - more (beyond the scope of this class)
- > Matlab conv2 uses zero-padding

WHAT WE WILL LEARN TODAY?

1. Image sampling and quantization
2. Image histograms
3. Images as functions
4. Linear systems (filters)
5. Convolution and correlation

Some background reading:
Forsyth and Ponce, Computer Vision, Chapter 7

Cross correlation of two 2D signals $f[n,m]$ and $g[n,m]$

$$r_{fg}[k, l] \triangleq \sum_{n=-\infty}^{\infty} \sum_{m=-\infty}^{\infty} f[n, m] g^*[n - k, m - l]$$

$$= \sum_{n=-\infty}^{\infty} \sum_{m=-\infty}^{\infty} f[n + k, m + l] g^*[n, m], \quad k, l \in \mathbb{Z}.$$

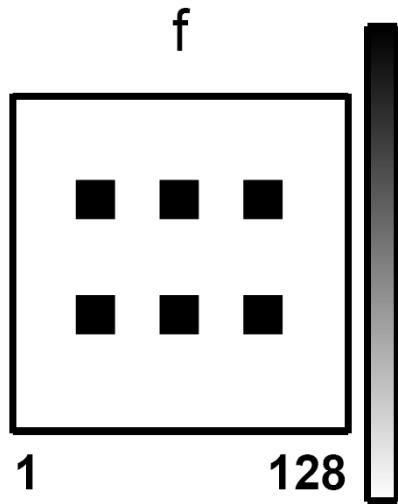
(k, l) is called the **lag**

- Equivalent to a convolution without the flip

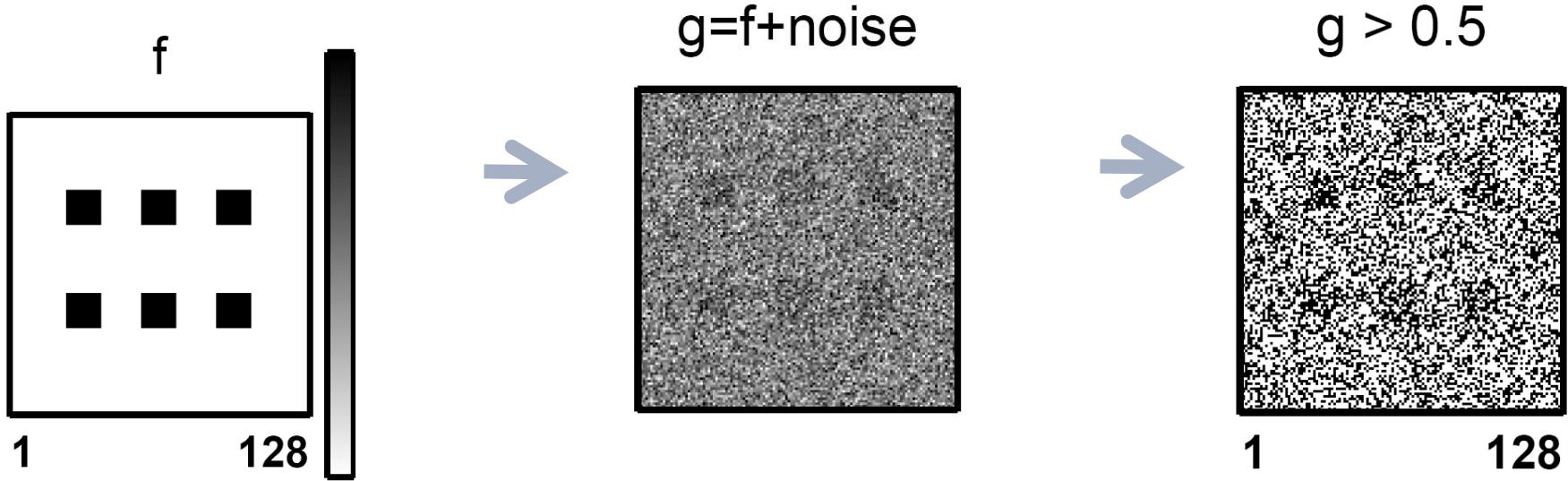
$$r_{fg}[n, m] = f[n, m] * g^*[-n, -m]$$

(g^* is defined as the *complex conjugate* of g . In this class, $g(n,m)$ are real numbers, hence $g^*=g$.)

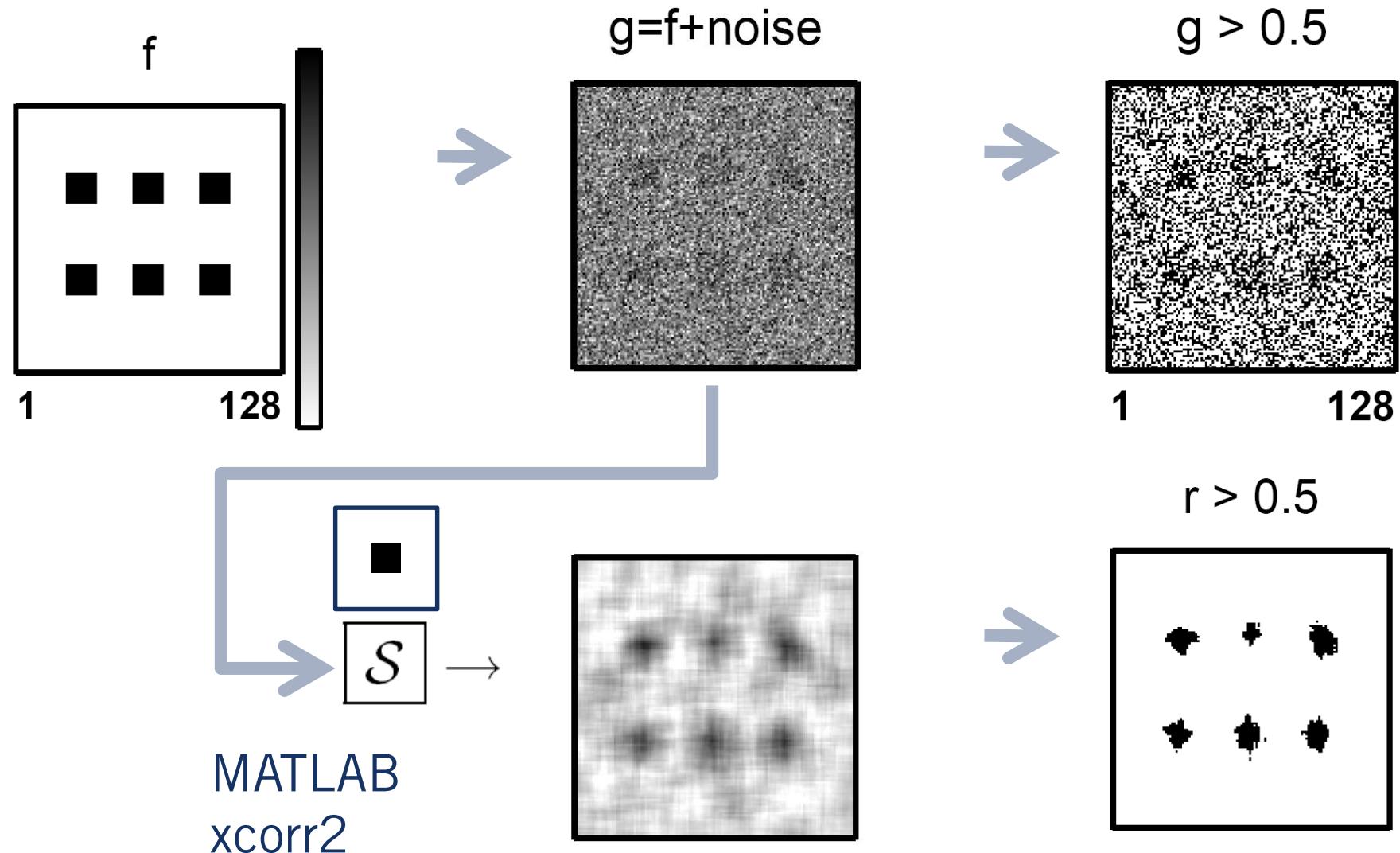
(CROSS) CORRELATION – EXAMPLE



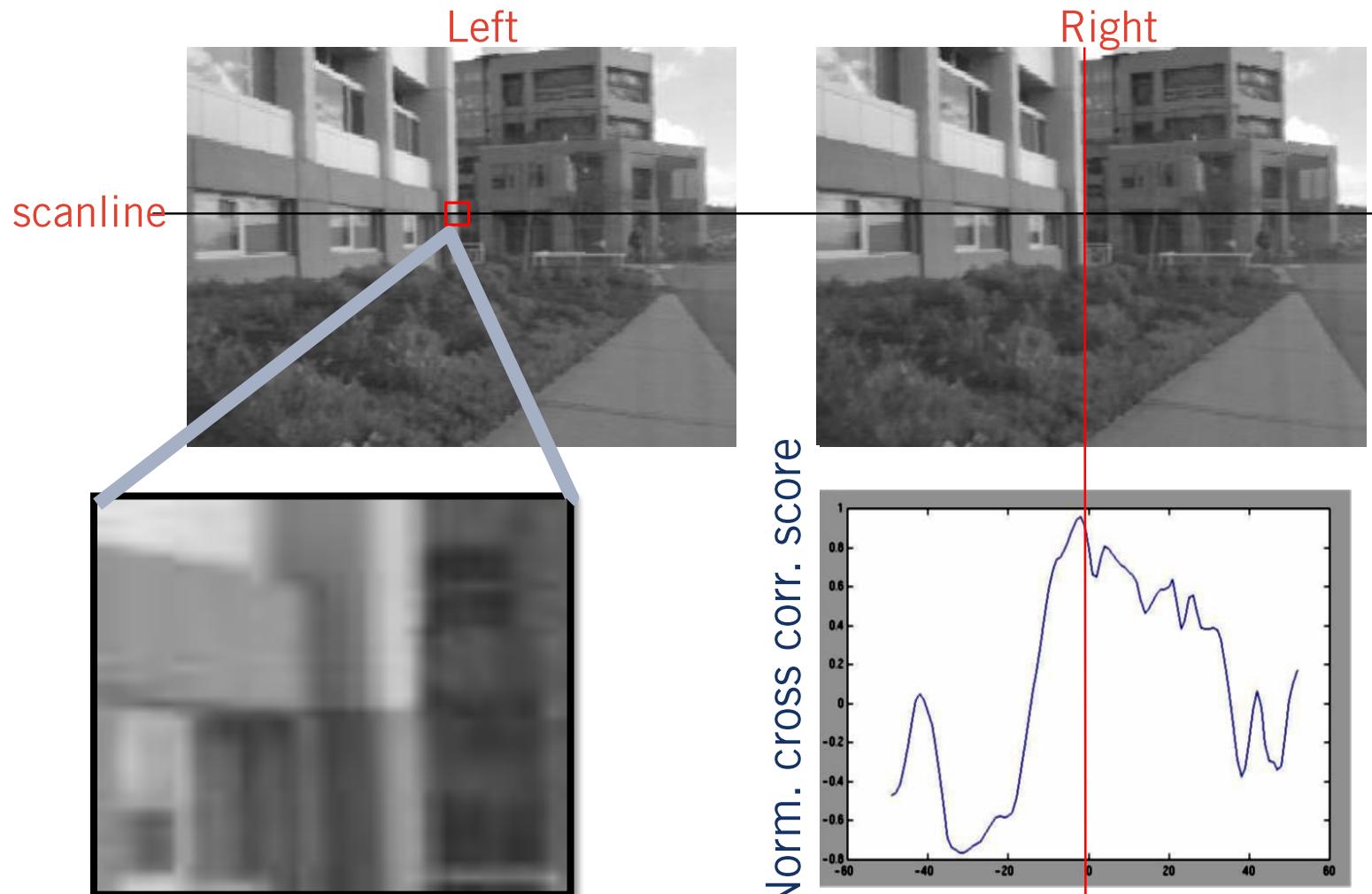
(CROSS) CORRELATION – EXAMPLE



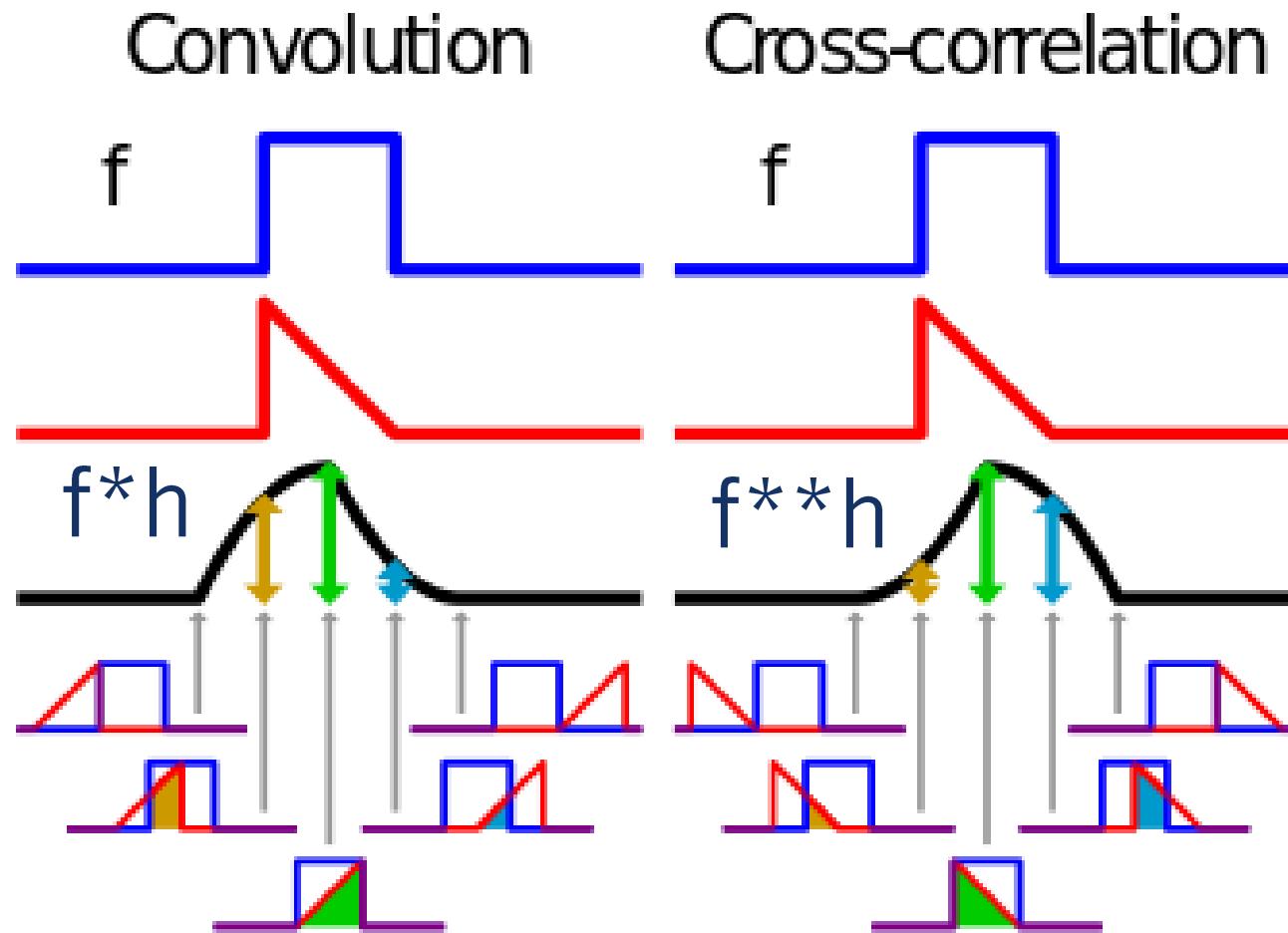
(CROSS) CORRELATION – EXAMPLE



(CROSS) CORRELATION – EXAMPLE



$$dc(y_1, y_2) = \frac{y_1^T y_2}{\|y_1\| \|y_2\|}$$



Cross Correlation Application: Vision system for TV remote control

- uses template matching

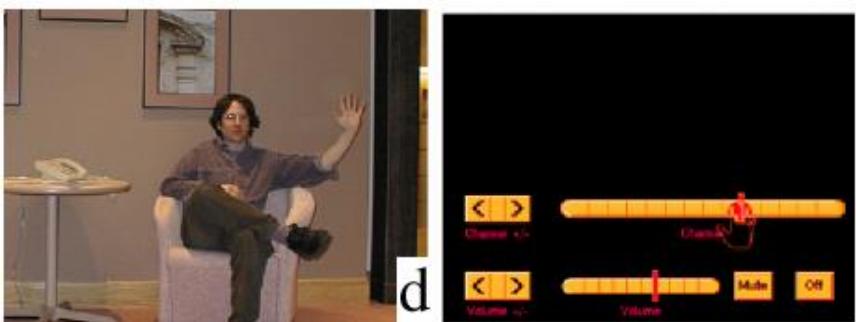
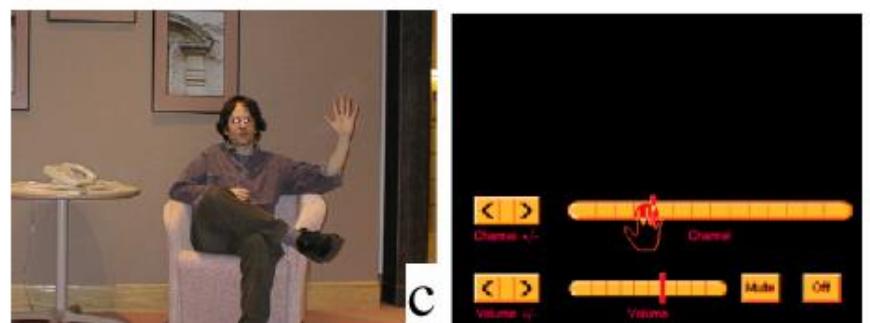
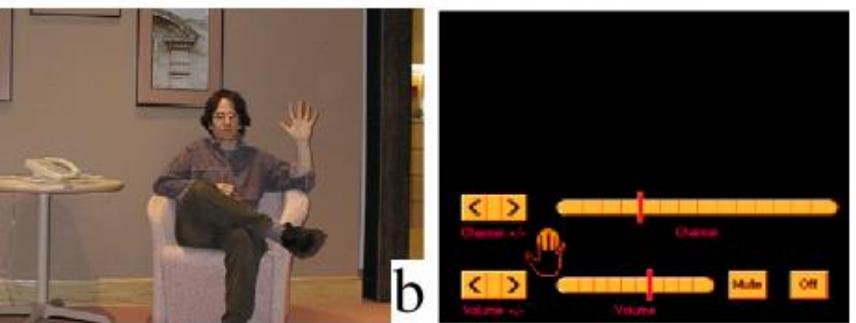


Figure from “Computer Vision for Interactive Computer Graphics,” W.Freeman et al, IEEE Computer Graphics and Applications, 1998 copyright 1998, IEEE

- **Associative property:**

$$(f \ast\ast h_1) \ast\ast h_2 = f \ast\ast (h_1 \ast\ast h_2)$$

- **Distributive property:**

$$f \ast\ast (h_1 + h_2) = (f \ast\ast h_1) + (f \ast\ast h_2)$$

The order doesn't matter! $h_1 \ast\ast h_2 = h_2 \ast\ast h_1$

- **Shift property:**

$$f[n, m] \ast\ast \delta_2[n - n_0, m - m_0] = f[n - n_0, m - m_0]$$

- **Shift-invariance:**

$$g[n, m] = f[n, m] \ast\ast h[n, m]$$

$$\implies f[n - l_1, m - l_1] \ast\ast h[n - l_2, m - l_2]$$

$$= g[n - l_1 - l_2, m - l_1 - l_2]$$

CONVOLUTION VS. (CROSS) CORRELATION

1. A **convolution** is an integral that expresses the amount of overlap of one function as it is shifted over another function.
→ convolution is a **filtering** operation

1. **Correlation** compares the *similarity* of **two** sets of **data**. Correlation computes a measure of similarity of two input signals as they are shifted by one another. The correlation result reaches a maximum at the time when the two signals match best.
→ correlation is a measure of **relatedness** of two signals

WHAT WE HAVE LEARNED TODAY?

1. Image sampling and quantization
2. Image histograms
3. Images as functions
4. Linear systems (filters)
5. Convolution and correlation

ANY QUESTIONS?

