

# Lecture 4

## Derivatives and edges

# Administrative

A1 is out

- It is graded
- Due **Jan 24**

# Administrative

Recitations (2 options)

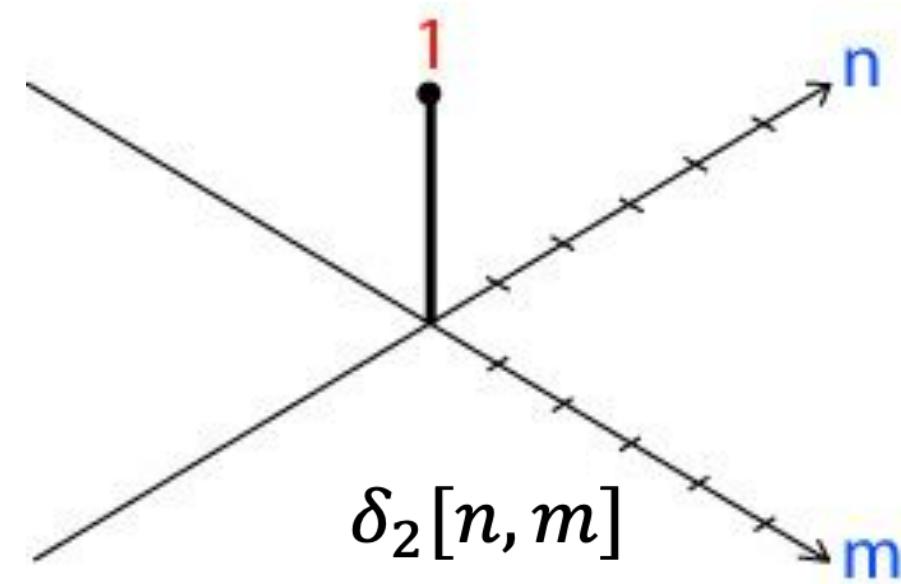
- Friday mornings 9:30-10:20am @ MGH 231
- Friday afternoons 12:30-1:20pm @ CSE2 G01

This week:

We will go over Python & Numpy basics

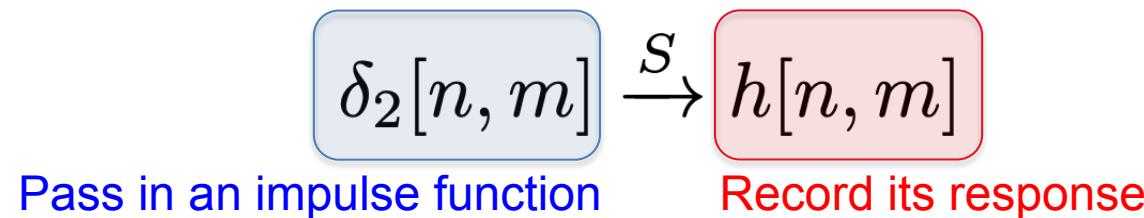
# So far: 2D impulse function

- A special function
- 1 at the origin [0,0].
- 0 everywhere else



So far: We get the **impulse response** when we pass an **impulse function** through a LSI system

- The moving average filter equation again:  $g[n, m] = \frac{1}{9} \sum_{k=-1}^1 \sum_{l=-1}^1 f[n - k, m - l]$



$$h[n, m] = \begin{bmatrix} 1/9 & 1/9 & 1/9 \\ 1/9 & 1/9 & 1/9 \\ 1/9 & 1/9 & 1/9 \end{bmatrix}$$

# So far: write down $f$ as a sum of impulses

Let's say our input  $f$  is a 3x3 image:

$$\begin{array}{|c|c|c|} \hline f[0,0] & f[0,1] & f[1,1] \\ \hline f[1,0] & f[1,1] & f[1,2] \\ \hline f[2,0] & f[2,1] & f[2,2] \\ \hline \end{array} = \begin{array}{|c|c|c|} \hline f[0,0] & 0 & 0 \\ \hline 0 & 0 & 0 \\ \hline 0 & 0 & 0 \\ \hline \end{array} + \begin{array}{|c|c|c|} \hline 0 & f[0,1] & 0 \\ \hline 0 & 0 & 0 \\ \hline 0 & 0 & 0 \\ \hline \end{array} + \dots + \begin{array}{|c|c|c|} \hline 0 & 0 & 0 \\ \hline 0 & 0 & 0 \\ \hline 0 & 0 & f[2,2] \\ \hline \end{array}$$
$$= f[0,0] \times \begin{array}{|c|c|c|} \hline 1 & 0 & 0 \\ \hline 0 & 0 & 0 \\ \hline 0 & 0 & 0 \\ \hline \end{array} + f[0,1] \times \begin{array}{|c|c|c|} \hline 0 & 1 & 0 \\ \hline 0 & 0 & 0 \\ \hline 0 & 0 & 0 \\ \hline \end{array} + \dots + f[2,2] \times \begin{array}{|c|c|c|} \hline 0 & 0 & 0 \\ \hline 0 & 0 & 0 \\ \hline 0 & 0 & 1 \\ \hline \end{array}$$
$$= f[0,0] \cdot \delta_2[n, m] + f[0,1] \cdot \delta_2[n, m - 1] + \dots + f[2,2] \cdot \delta_2[n - 2, m - 2]$$

# So far: We derived convolutions

- An LSI system is completely specified by its impulse response.
  - For any input  $f$ , we can compute the output  $g$  in terms of the impulse response  $h$ .

Discrete Convolution

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

So far: We created a sharpening system by combining filters

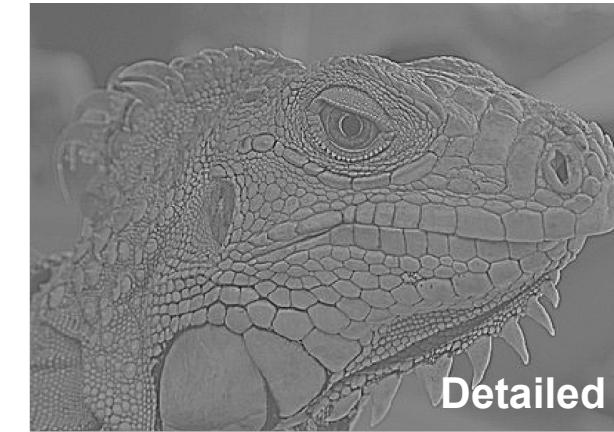


-



smoothed (3x3)

=

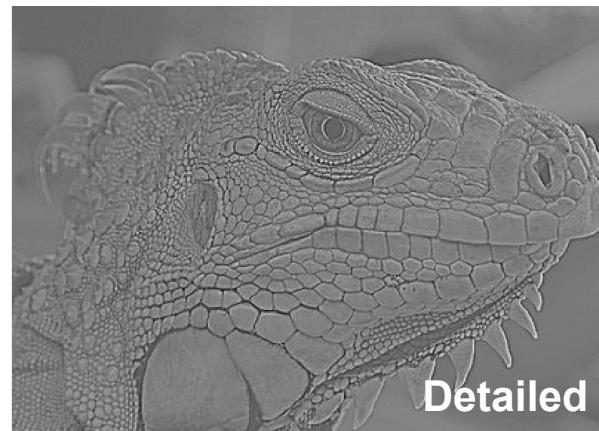


Detailed

Let's add it back to get a **sharpening system**:



+



Detailed

=



Sharpened

# (Cross) correlation – symbol: \*\*

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

$$f[n, m] \text{** } h[n, m] = \sum_k \sum_l f[k, l] h[n + k, m + l]$$

Equivalent to a convolution without the flip

# Today's agenda

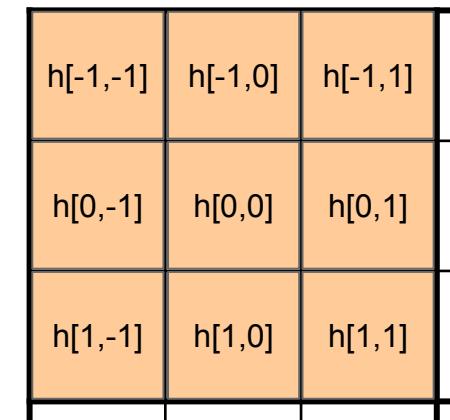
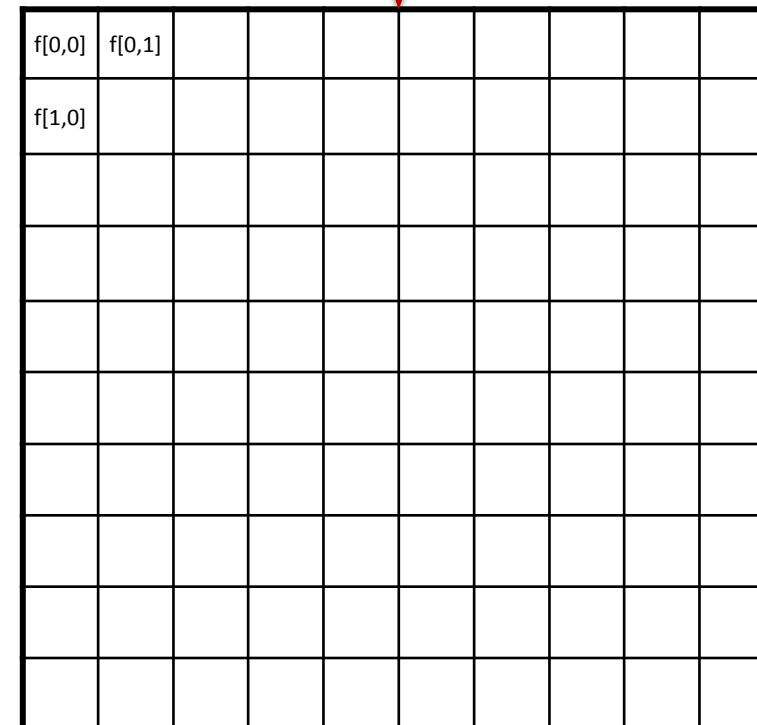
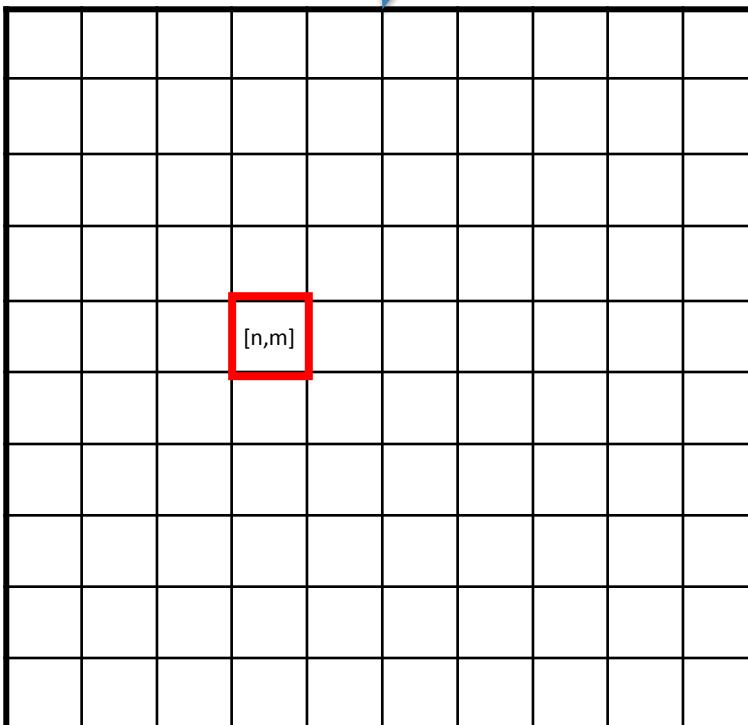
- Convolutions and Cross-Correlation
- Edge detection
- Image Gradients
- A simple edge detector

# Today's agenda

- Convolutions and Cross-Correlation
- Edge detection
- Image Gradients
- A simple edge detector

# 2D Discrete Convolution

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



Kernel  $h[k, l]$

# 2D Discrete Convolution

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

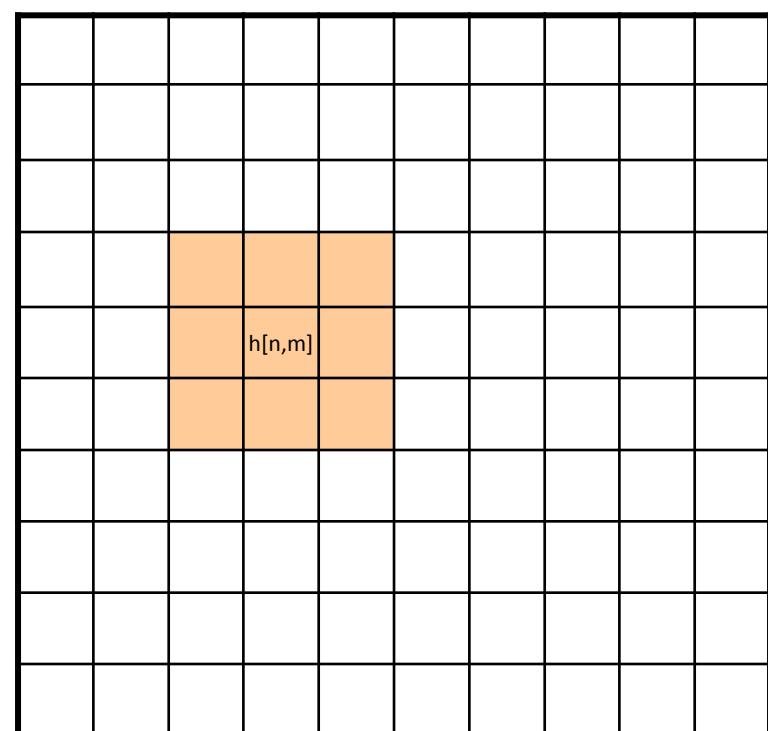
$h[-1, -1]$	$h[-1, 0]$	$h[-1, 1]$
$h[0, -1]$	$h[0, 0]$	$h[0, 1]$
$h[1, -1]$	$h[1, 0]$	$h[1, 1]$

Fold

$h[1, 1]$	$h[1, 0]$	$h[1, -1]$
$h[0, 1]$	$h[0, 0]$	$h[0, -1]$
$h[-1, 1]$	$h[-1, 0]$	$h[-1, -1]$

Kernel  $h[k, l]$

Shift



Kernel  $h[n, m]$

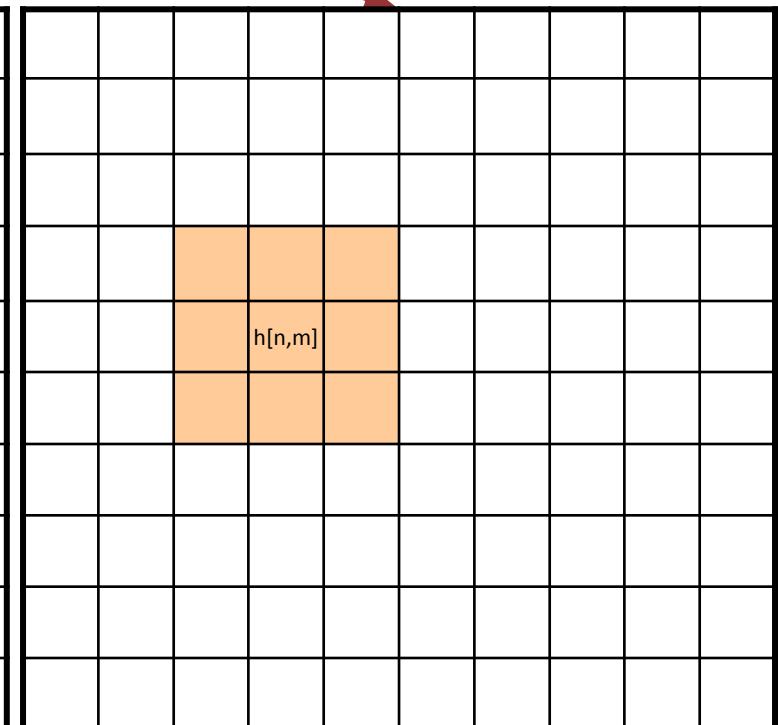
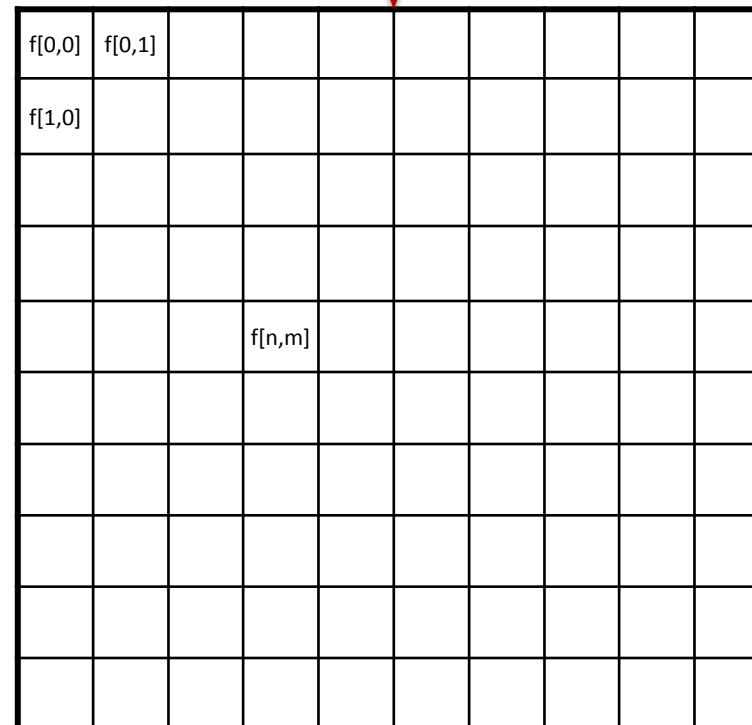
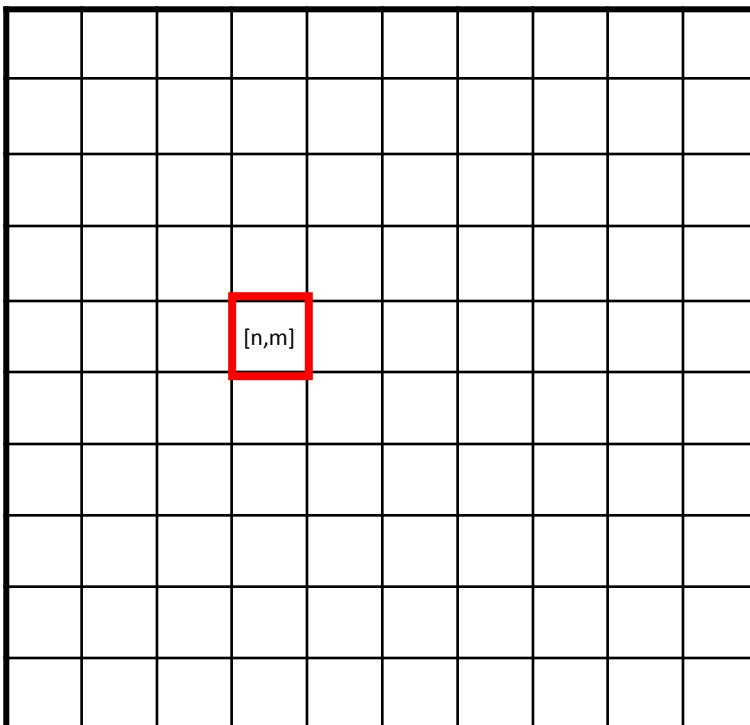
# 2D Discrete Convolution

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

Output  $f * h$

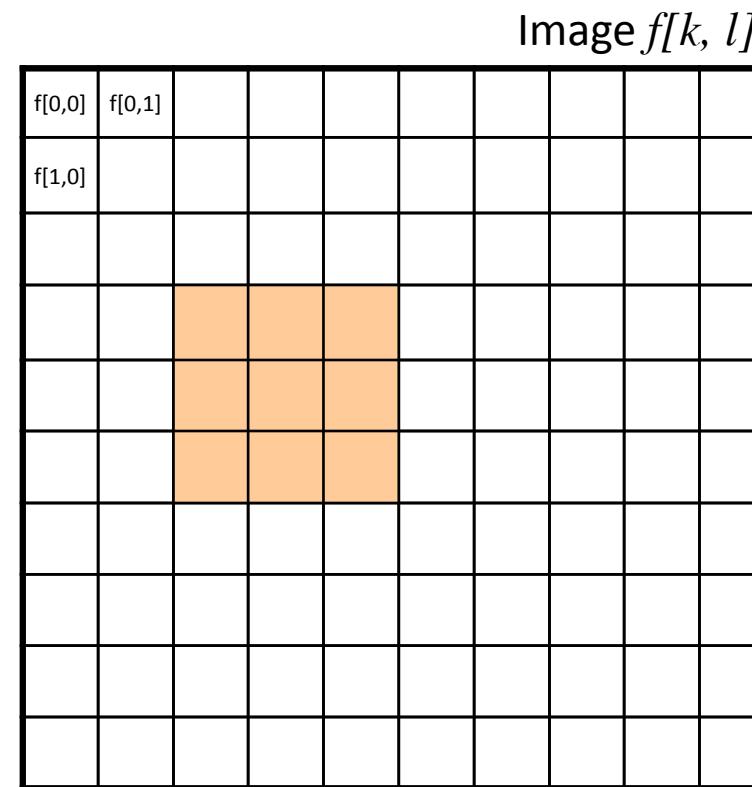
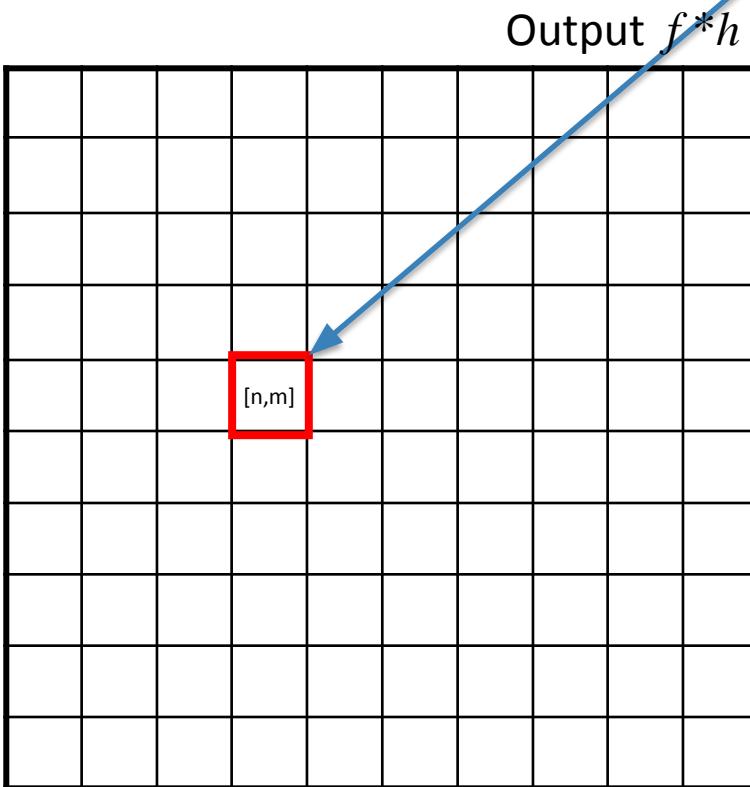
Image  $f[k, l]$

Kernel  $h[n-k, m-l]$



# 2D Discrete Convolution

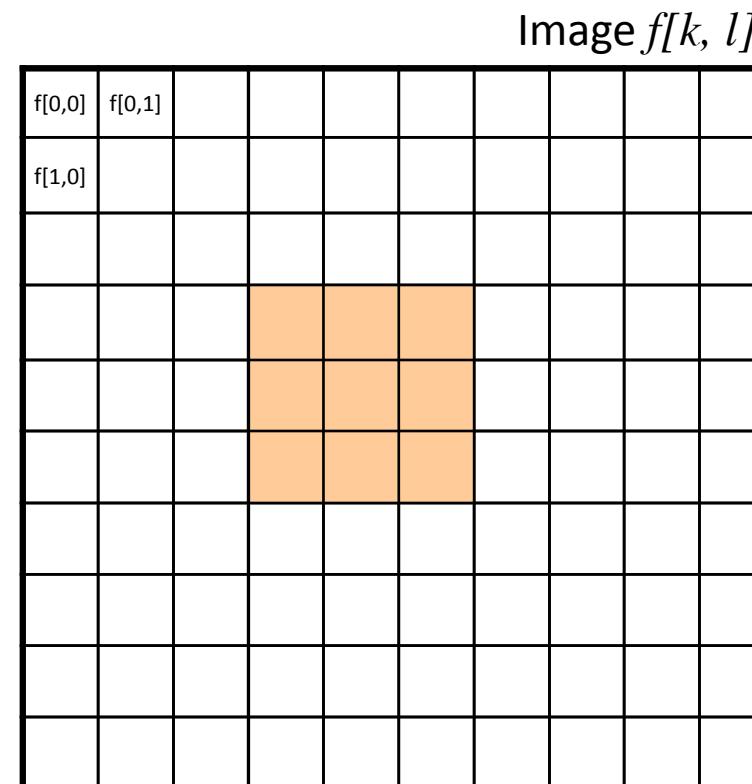
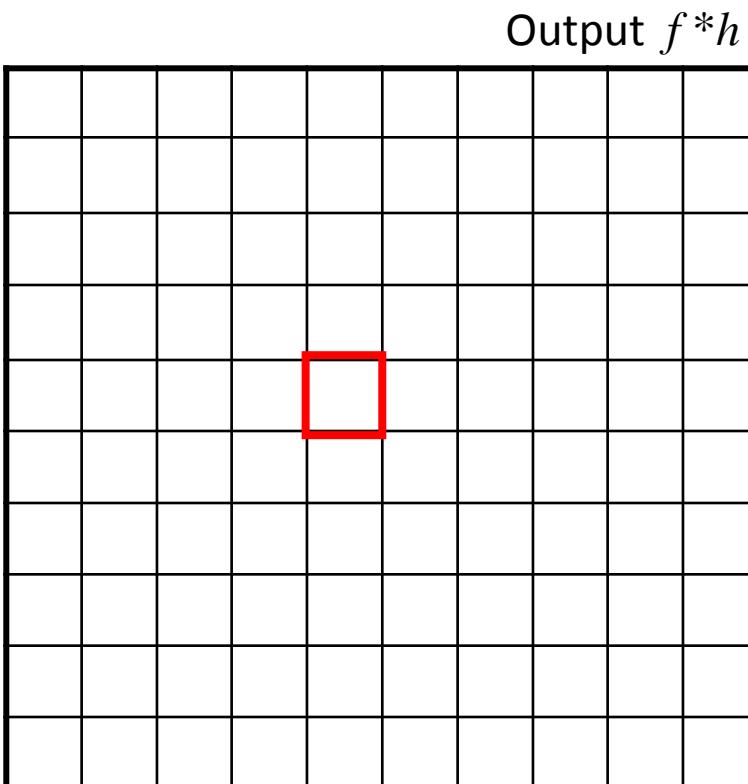
$$f[n, m] * h[n, m] = \sum_{k=-\infty}^{\infty} \sum_{l=-\infty}^{\infty} f[k, l] \cdot h[n - k, m - l]$$



Element-wise multiplication  
Image  $f[k, l] \bullet$  Kernel  $h[n-k, m-l]$

# 2D Discrete Convolution

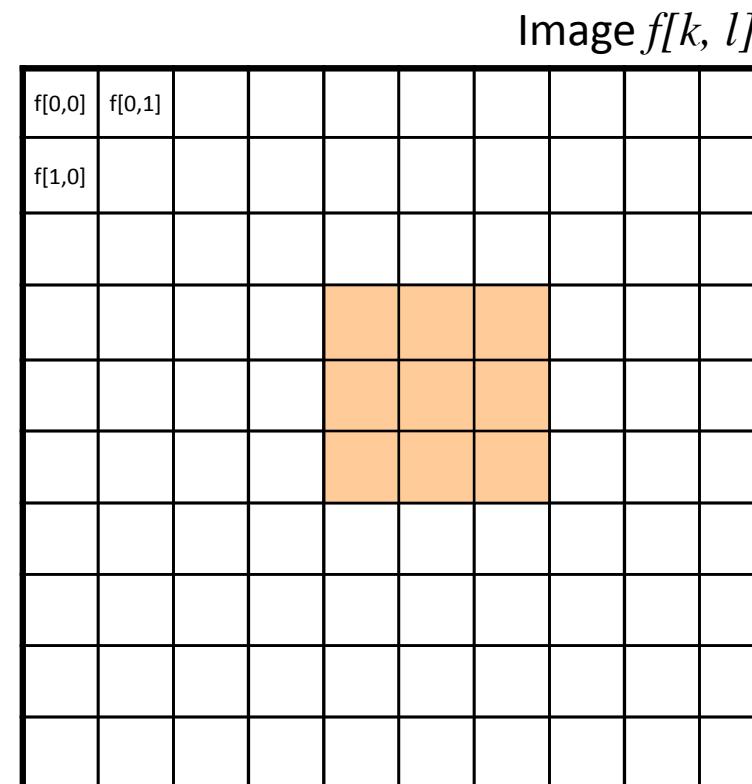
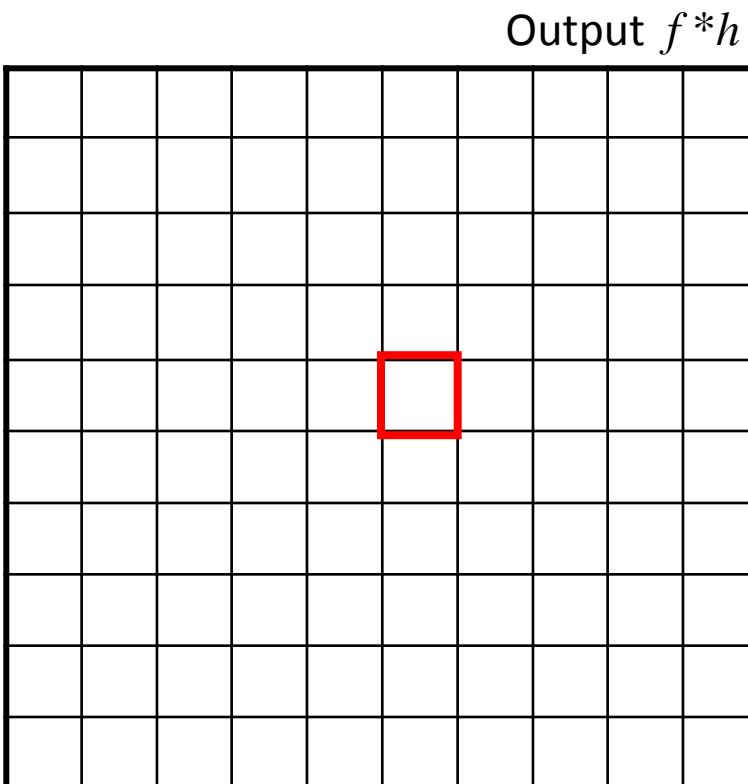
$$f[n, m] * h[n, m] = \sum_{k=-\infty}^{\infty} \sum_{l=-\infty}^{\infty} f[k, l] \cdot h[n - k, m - l]$$



Element-wise multiplication  
Image  $f[k, l] \bullet$  Kernel  $h[n-k, m-l]$

# 2D Discrete Convolution

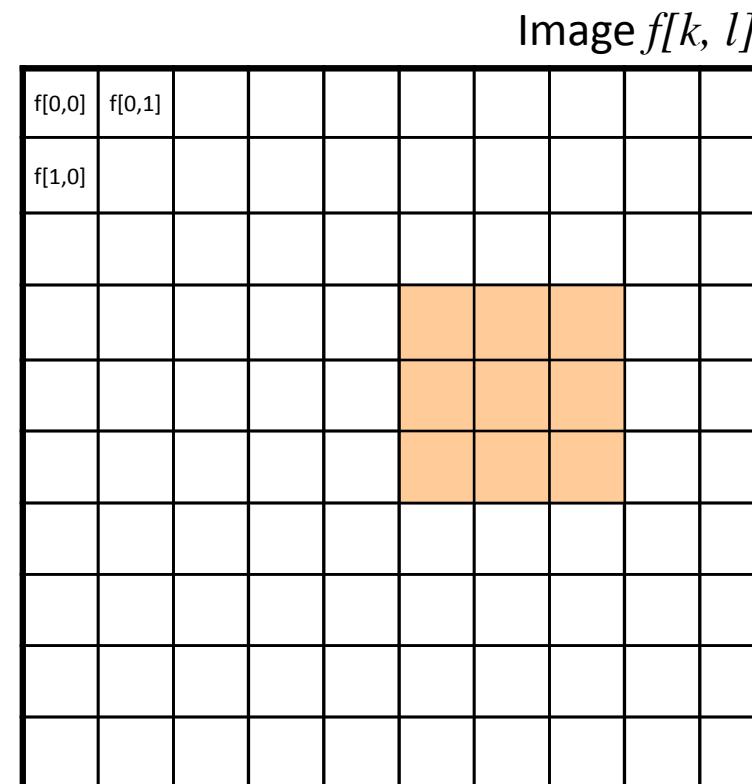
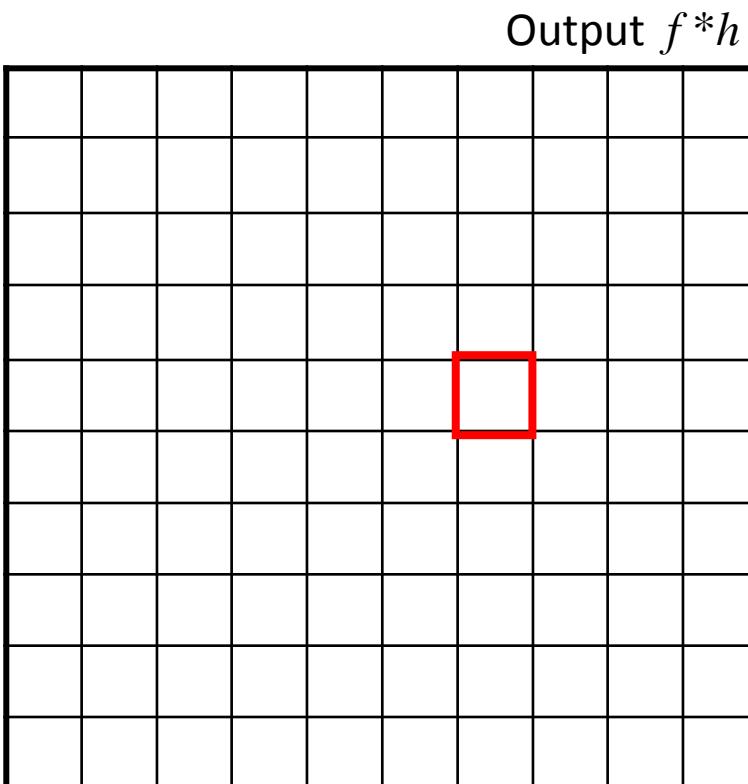
$$f[n, m] * h[n, m] = \sum_{k=-\infty}^{\infty} \sum_{l=-\infty}^{\infty} f[k, l] \cdot h[n - k, m - l]$$



Element-wise multiplication  
Image  $f[k, l] \bullet$  Kernel  $h[n-k, m-l]$

# 2D Discrete Convolution

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



Element-wise multiplication  
Image  $f[k, l] \bullet$  Kernel  $h[n-k, m-l]$

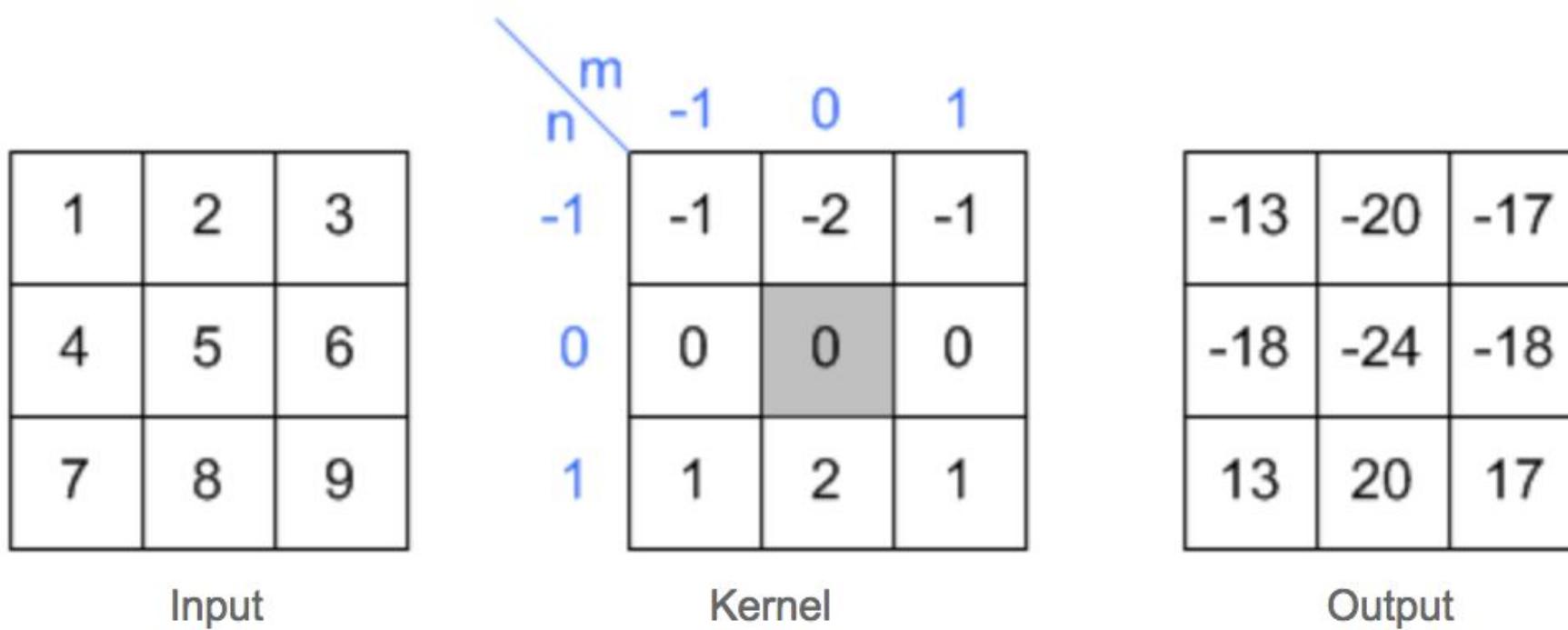
# 2D Discrete Convolution

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

## Algorithm:

- Fold  $h[k, l]$  about origin to form  $h[-k, -l]$
- Shift the folded results by  $n, m$  to form  $h[n - k, m - l]$
- Multiply  $h[n - k, m - l]$  by  $f[k, l]$
- Sum over all  $k, l$ , store result in output position  $[n, m]$
- Repeat for every  $n, m$

# 2D convolution example



Slide credit: Song Ho Ahn

# 2D convolution example

1	2	1
0	0	0
-1	-2	-1
7	8	9

$$\begin{aligned} &= x[-1,-1] \cdot h[1,1] + x[0,-1] \cdot h[0,1] + x[1,-1] \cdot h[-1,1] \\ &\quad + x[-1,0] \cdot h[1,0] + x[0,0] \cdot h[0,0] + x[1,0] \cdot h[-1,0] \\ &\quad + x[-1,1] \cdot h[1,-1] + x[0,1] \cdot h[0,-1] + x[1,1] \cdot h[-1,-1] \\ &= 0 \cdot 1 + 0 \cdot 2 + 0 \cdot 1 + 0 \cdot 0 + 1 \cdot 0 + 2 \cdot 0 + 0 \cdot (-1) + 4 \cdot (-2) + 5 \cdot (-1) = -13 \end{aligned}$$

-13	-20	-17
-18	-24	-18
13	20	17

Output

Slide credit: Song Ho Ahn

# 2D convolution example

1	2	1
0	0	0
1	2	3
-1	-2	-1
4	5	6

| 7 | 8 | 9 |

$$\begin{aligned} &= x[0,-1] \cdot h[1,1] + x[1,-1] \cdot h[0,1] + x[2,-1] \cdot h[-1,1] \\ &\quad + x[0,0] \cdot h[1,0] + x[1,0] \cdot h[0,0] + x[2,0] \cdot h[-1,0] \\ &\quad + x[0,1] \cdot h[1,-1] + x[1,1] \cdot h[0,-1] + x[2,1] \cdot h[-1,-1] \\ &= 0 \cdot 1 + 0 \cdot 2 + 0 \cdot 1 + 1 \cdot 0 + 2 \cdot 0 + 3 \cdot 0 + 4 \cdot (-1) + 5 \cdot (-2) + 6 \cdot (-1) = -20 \end{aligned}$$

-13	-20	-17
-18	-24	-18
13	20	17

Output

Slide credit: Song Ho Ahn

# 2D convolution example

	1	2	1
1	0	0	0
4	-1	5	-2
7	8	9	

$$\begin{aligned} &= x[1,-1] \cdot h[1,1] + x[2,-1] \cdot h[0,1] + x[3,-1] \cdot h[-1,1] \\ &\quad + x[1,0] \cdot h[1,0] + x[2,0] \cdot h[0,0] + x[3,0] \cdot h[-1,0] \\ &\quad + x[1,1] \cdot h[1,-1] + x[2,1] \cdot h[0,-1] + x[3,1] \cdot h[-1,-1] \\ &= 0 \cdot 1 + 0 \cdot 2 + 0 \cdot 1 + 2 \cdot 0 + 3 \cdot 0 + 0 \cdot 0 + 5 \cdot (-1) + 6 \cdot (-2) + 0 \cdot (-1) = -17 \end{aligned}$$

-13	-20	-17
-18	-24	-18
13	20	17

Output

Slide credit: Song Ho Ahn

# 2D convolution example

1	2	1	2	3
0	0	0	5	6
-1	-2	-1	8	9

$$\begin{aligned} &= x[-1,0] \cdot h[1,1] + x[0,0] \cdot h[0,1] + x[1,0] \cdot h[-1,1] \\ &\quad + x[-1,1] \cdot h[1,0] + x[0,1] \cdot h[0,0] + x[1,1] \cdot h[-1,0] \\ &\quad + x[-1,2] \cdot h[1,-1] + x[0,2] \cdot h[0,-1] + x[1,2] \cdot h[-1,-1] \\ &= 0 \cdot 1 + 1 \cdot 2 + 2 \cdot 1 + 0 \cdot 0 + 4 \cdot 0 + 5 \cdot 0 + 0 \cdot (-1) + 7 \cdot (-2) + 8 \cdot (-1) = -18 \end{aligned}$$

-13	-20	-17
-18	-24	-18
13	20	17

Output

Slide credit: Song Ho Ahn

# 2D convolution example

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

$$\begin{aligned} &= x[0,0] \cdot h[1,1] + x[1,0] \cdot h[0,1] + x[2,0] \cdot h[-1,1] \\ &\quad + x[0,1] \cdot h[1,0] + x[1,1] \cdot h[0,0] + x[2,1] \cdot h[-1,0] \\ &\quad + x[0,2] \cdot h[1,-1] + x[1,2] \cdot h[0,-1] + x[2,2] \cdot h[-1,-1] \\ &= 1 \cdot 1 + 2 \cdot 2 + 3 \cdot 1 + 4 \cdot 0 + 5 \cdot 0 + 6 \cdot 0 + 7 \cdot (-1) + 8 \cdot (-2) + 9 \cdot (-1) = -24 \end{aligned}$$

-13	-20	-17
-18	-24	-18
13	20	17

Output

Slide credit: Song Ho Ahn

# 2D convolution example

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

$$\begin{aligned} &= x[1,0] \cdot h[1,1] + x[2,0] \cdot h[0,1] + x[3,0] \cdot h[-1,1] \\ &\quad + x[1,1] \cdot h[1,0] + x[2,1] \cdot h[0,0] + x[3,1] \cdot h[-1,0] \\ &\quad + x[1,2] \cdot h[1,-1] + x[2,2] \cdot h[0,-1] + x[3,2] \cdot h[-1,-1] \\ &= 2 \cdot 1 + 3 \cdot 2 + 0 \cdot 1 + 5 \cdot 0 + 6 \cdot 0 + 0 \cdot 0 + 8 \cdot (-1) + 9 \cdot (-2) + 0 \cdot (-1) = -18 \end{aligned}$$

-13	-20	-17
-18	-24	-18
13	20	17

Output

# Practice with convolution


$$\text{Original} \quad * \quad \begin{array}{|c|c|c|} \hline 0 & 0 & 0 \\ \hline 0 & 1 & 0 \\ \hline 0 & 0 & 0 \\ \hline \end{array} = ?$$

# Practice with convolution

Original

\*

0	0	0
0	1	0
0	0	0

=

Filtered  
(no change)

# Practice with convolution


$$\text{Original} \quad * \quad \begin{array}{|c|c|c|} \hline 0 & 0 & 0 \\ \hline 0 & 0 & 1 \\ \hline 0 & 0 & 0 \\ \hline \end{array} = ?$$

# Practice with convolution

The diagram illustrates a convolution operation. On the left is a grayscale image of an eye, labeled "Original". In the center is a 3x3 kernel matrix with values:

0	0	0
0	0	1
0	0	0

To the right of the kernel is an equals sign followed by another grayscale image of an eye, labeled "Shifted right By 1 pixel".

# Practice with convolution



Original

$$\text{Original} * \frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} = ?$$

# Practice with convolution

$$\text{Original} \quad * \frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} = \text{Blurry output}$$

# What happens if a system contains multiple filters?



Original

0	0	0
0	2	0
0	0	0

-

$\frac{1}{9}$	1	1	1
1	1	1	1
1	1	1	1

= ?

(Note that filter sums to 1)

# What happens if a system contains multiple filters?



Original

0	0	0
0	2	0
0	0	0

-

$\frac{1}{9}$	1	1	1
1	1	1	1
1	1	1	1

=

0	0	0
0	1	0
0	0	0

+

0	0	0
0	1	0
0	0	0

-

$\frac{1}{9}$	1	1	1
1	1	1	1
1	1	1	1

# What does blurring take away?

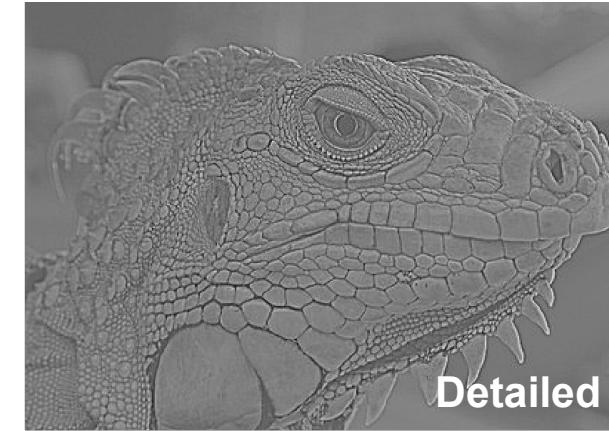


-



smoothed (3x3)

=



Detailed

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

# What does blurring take away?



-



=



Let's add it back to get a **sharpening system**:



+



=



# Convolution in 2D – Sharpening filter



Original

Sharpening system



**Sharpening system:** Accentuates differences with local average

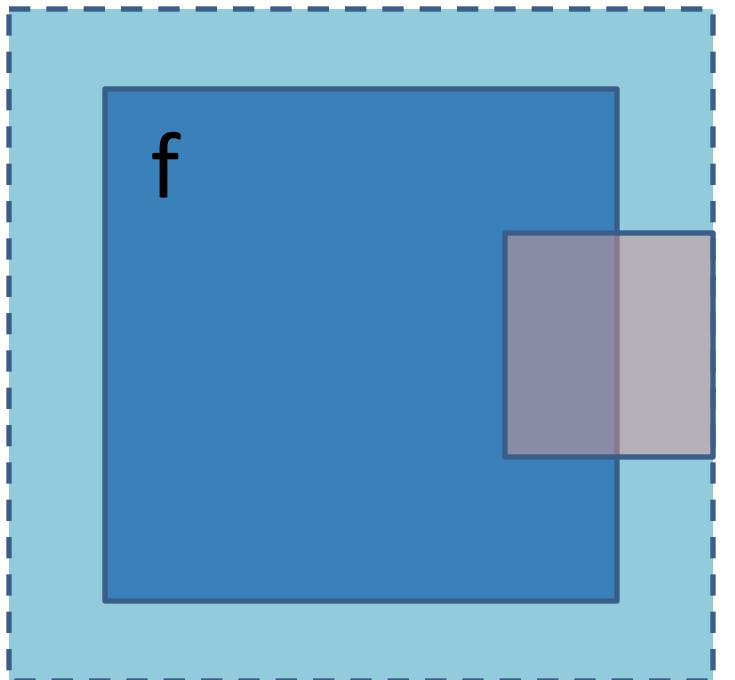
# Implementation detail: Image support and edge effect

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

$$\begin{array}{c} \text{Blue square} \\ \text{N}_1 \times \text{M}_1 \end{array} * \begin{array}{c} \text{Red square} \\ \text{N}_2 \times \text{M}_2 \end{array} = \begin{array}{c} \text{Large green L-shaped block} \\ (\text{N}_1 + \text{N}_2 - 1) \times (\text{M}_1 + \text{M}_2 - 1) \end{array}$$

# Image support and edge effect

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



h

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

# Today's agenda

- Convolutions and Cross-Correlation
- Edge detection
- Image Gradients
- A simple edge detector

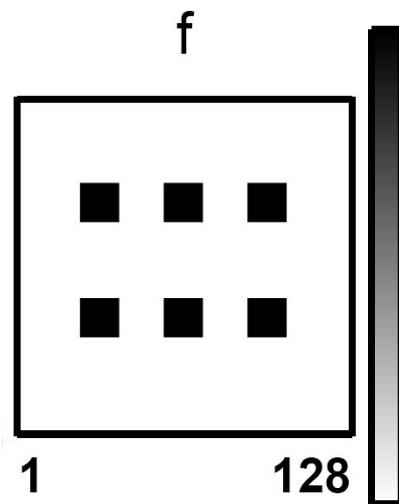
# (Cross) correlation – symbol: \*\*

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

$$f[n, m] \text{** } h[n, m] = \sum_k \sum_l f[k, l] h[n + k, m + l]$$

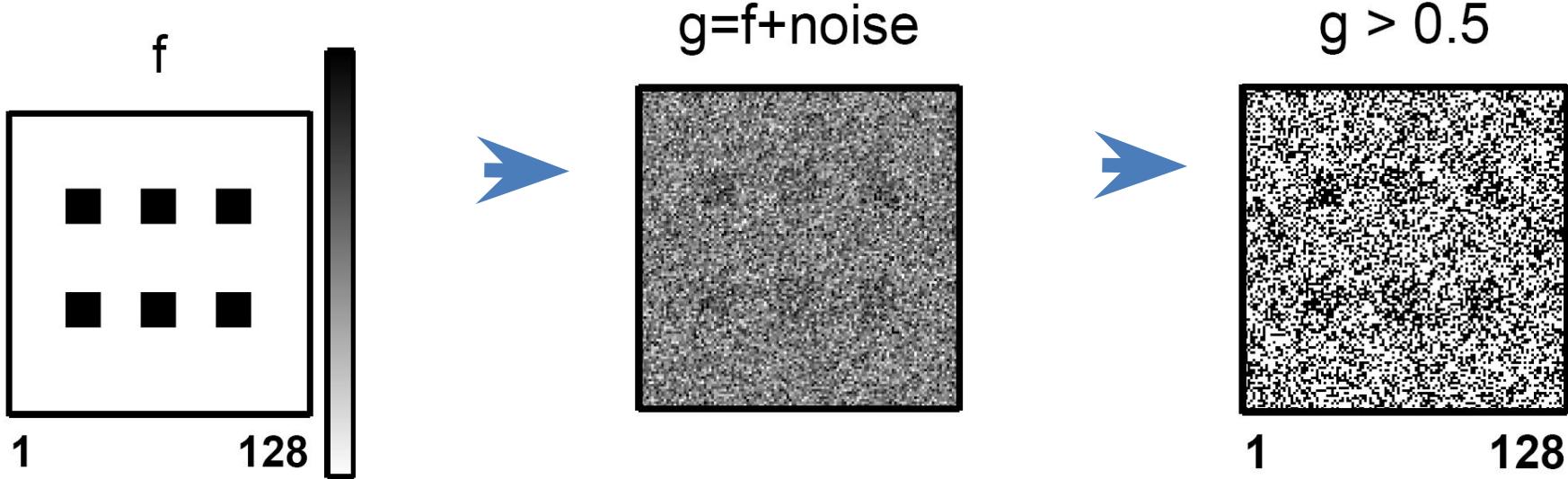
- Equivalent to a convolution without the flip
- Use it to measure ‘similarity’ between  $f$  and  $h$ .

# (Cross) correlation – example



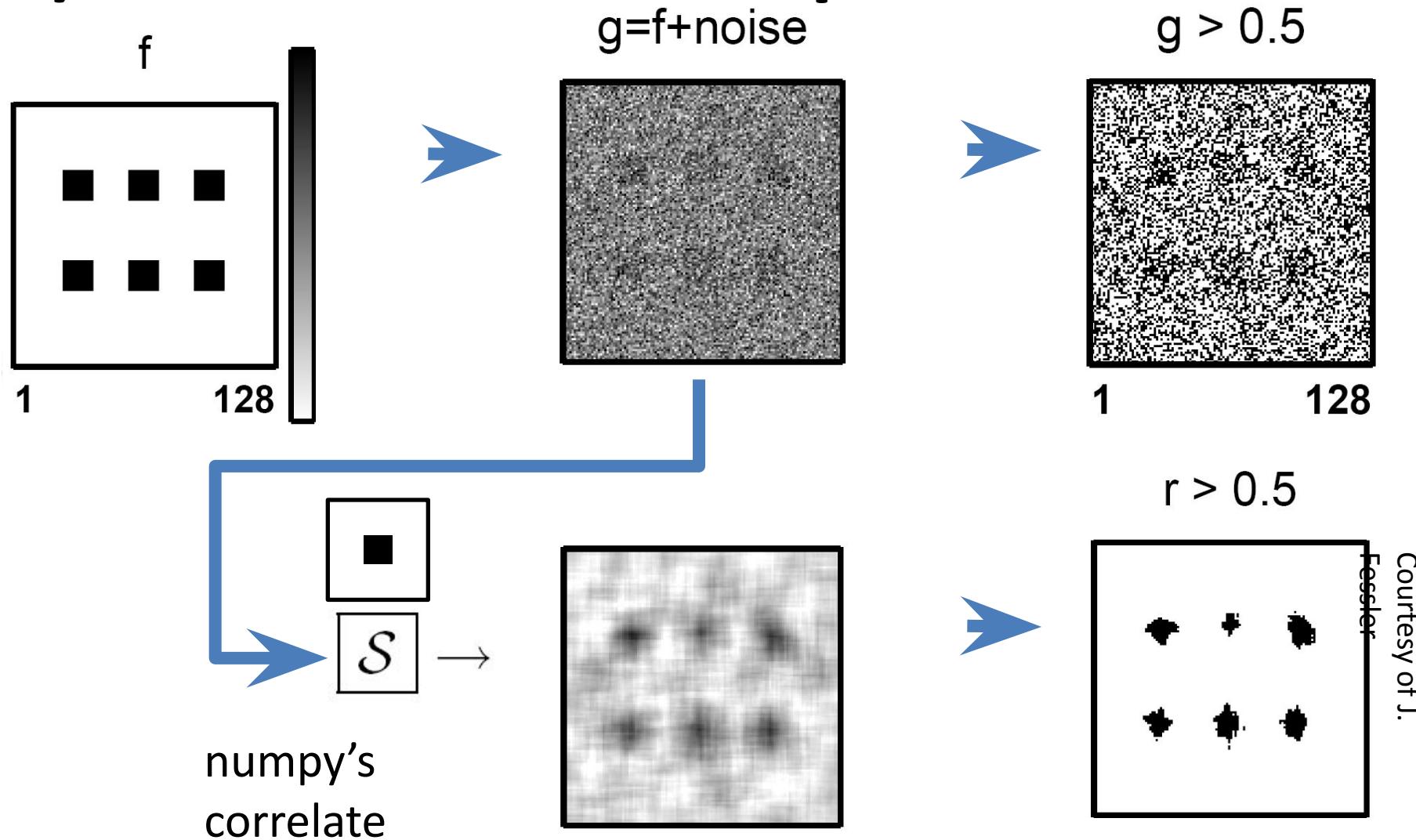
Courtesy of J.  
Fessler

# (Cross) correlation – example



Courtesy of J.  
Fessler

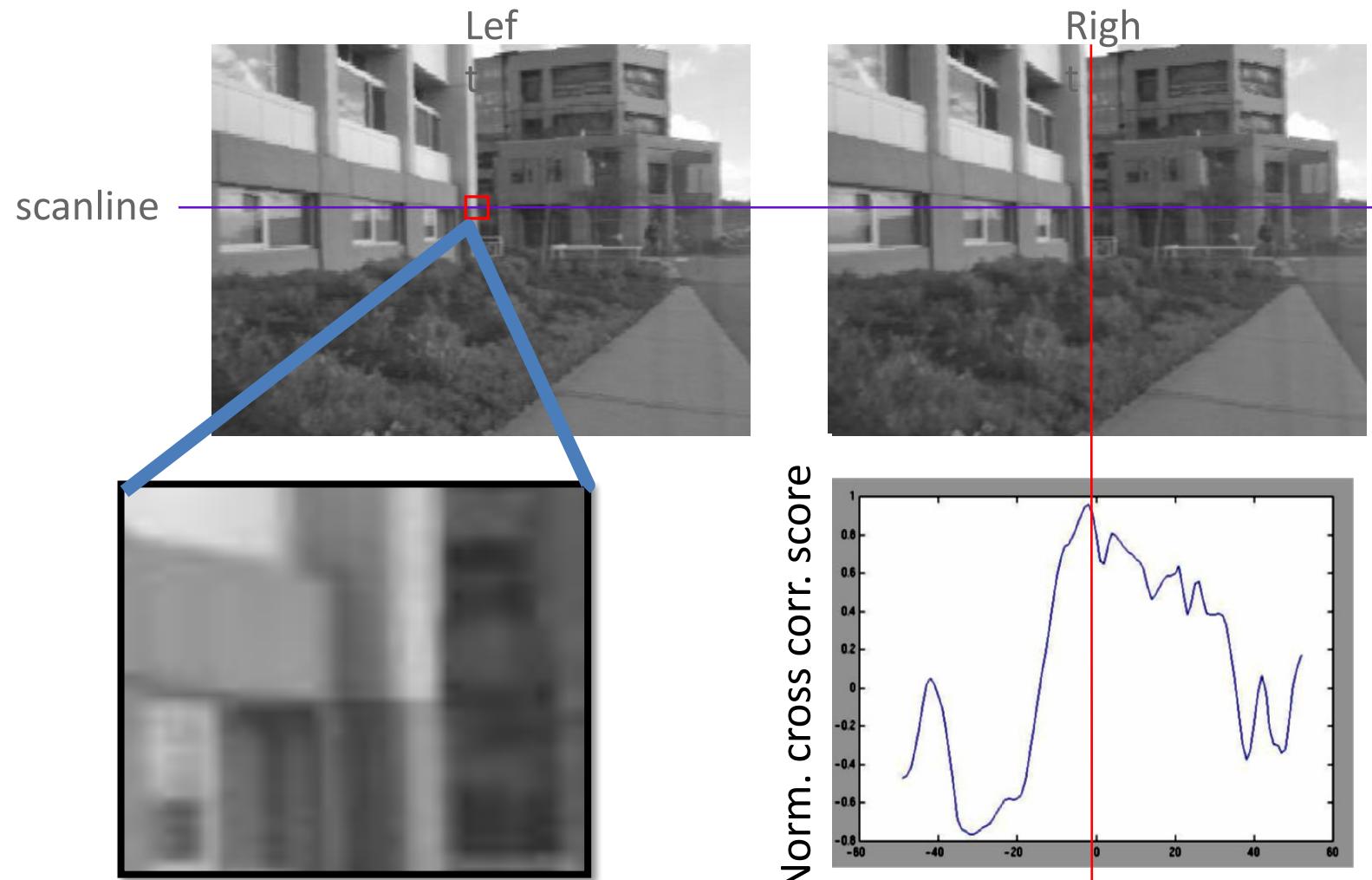
# (Cross) correlation – example



Courtesy of J.  
Fessler

Courtesy of J.  
Fessler

# (Cross) correlation – example





## Cross Correlation Application: Vision system for TV remote control

- uses template matching

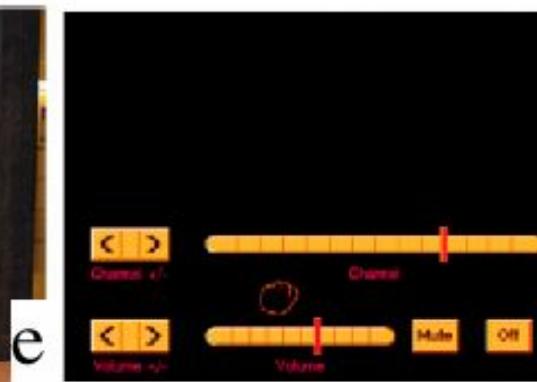
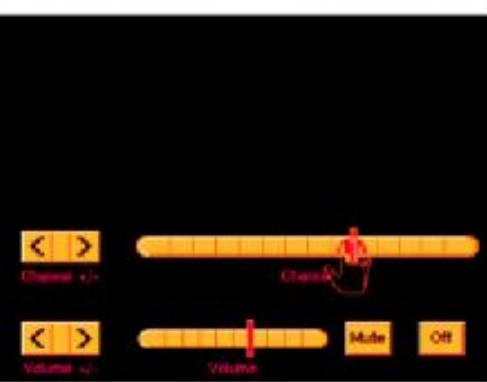
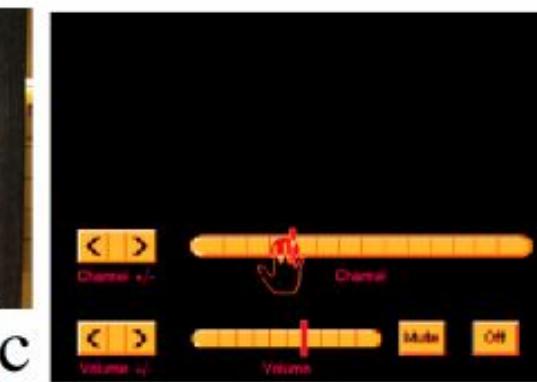
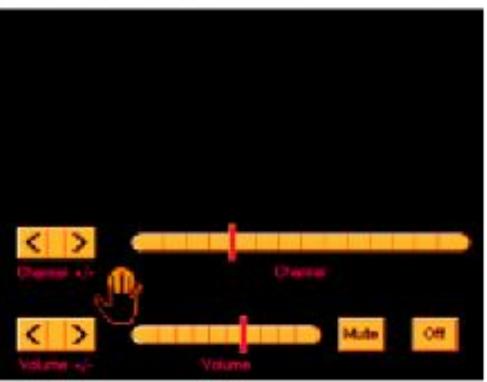


Figure from "Computer Vision for Interactive Computer Graphics," W.Freeman et al, IEEE Computer Graphics and Applications, 1998 copyright 1998, IEEE

# Properties of cross correlation

- Associative property:

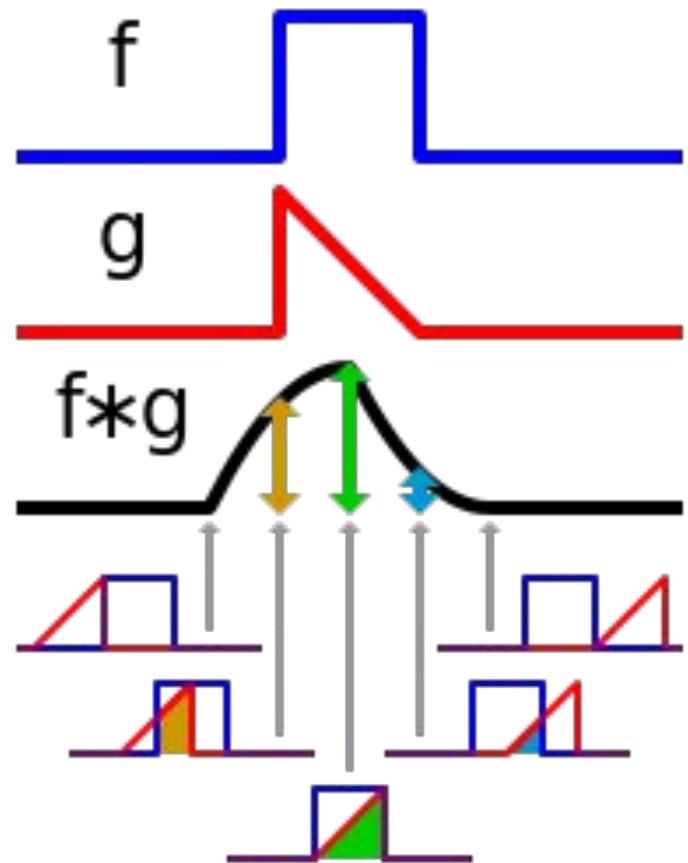
$$(f \ast\ast h_1) \ast\ast h_2 = f \ast\ast (h_1 \ast\ast h_2)$$

- Distributive property:

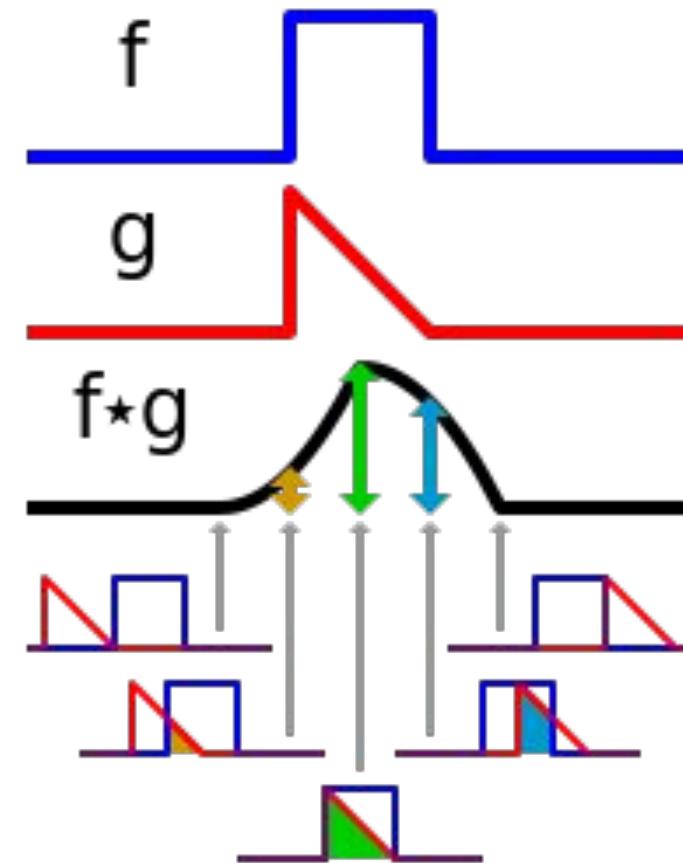
$$f \ast\ast (h_1 + h_2) = (f \ast\ast h_1) + (f \ast\ast h_2)$$

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

Convolution



Cross-correlation



# Convolution vs. (Cross) Correlation

- When is correlation equivalent to convolution?
- In other words, Q. when is  $f^{**}g = f^*g$ ?

# Convolution vs. (Cross) Correlation

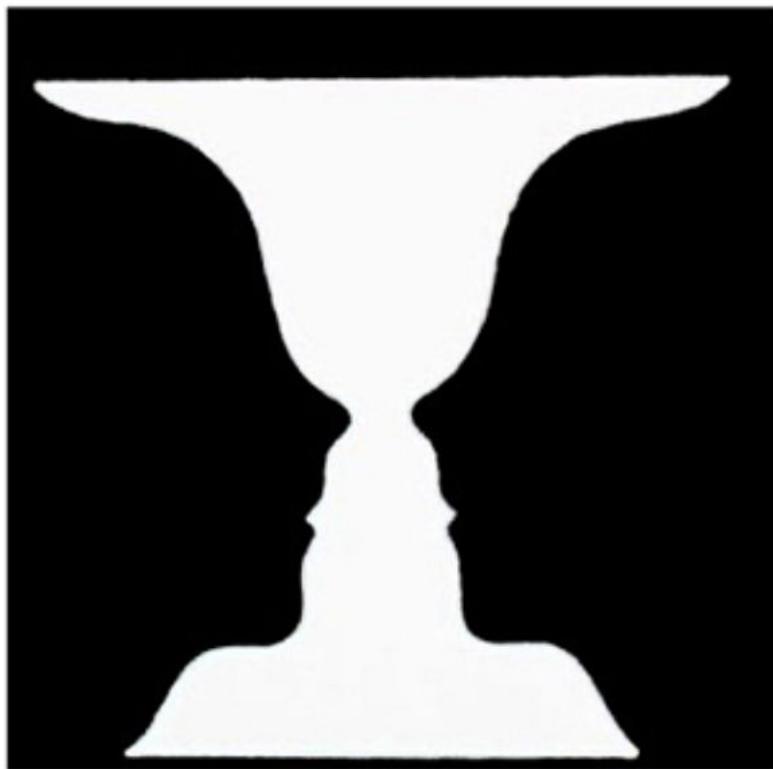
- A **convolution** is an integral that expresses the amount of overlap of one function as it is shifted over another function.
  - convolution is a **filtering** operation
- **Correlation** compares the ***similarity of two sets of data***. Correlation computes a measure of similarity of two input signals 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 will learn today

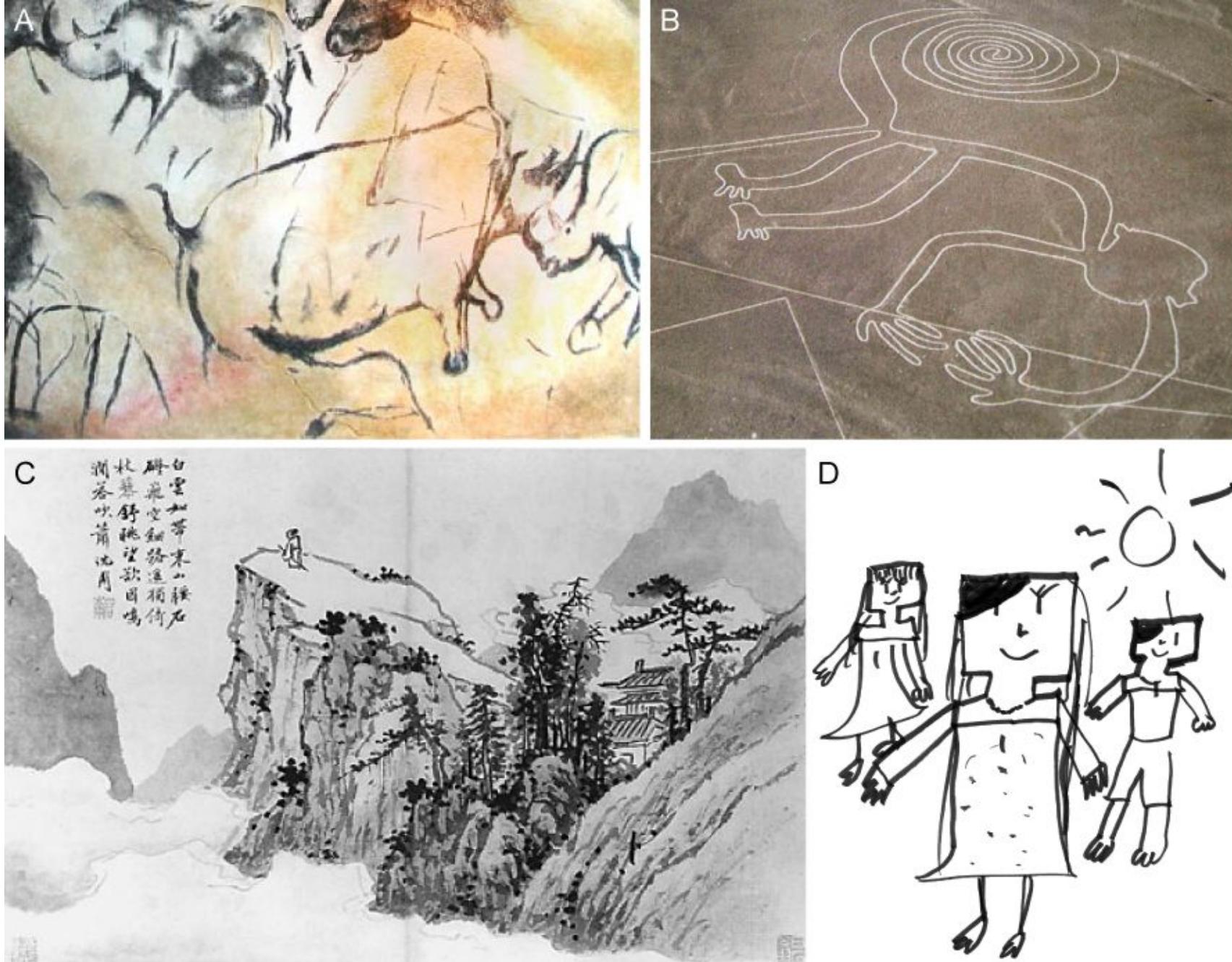
- Convolutions and Cross-Correlation
- Edge detection
- Image Gradients
- A simple edge detector

Some background reading:  
Forsyth and Ponce, Computer Vision, Chapter 8

Q. What do you see?

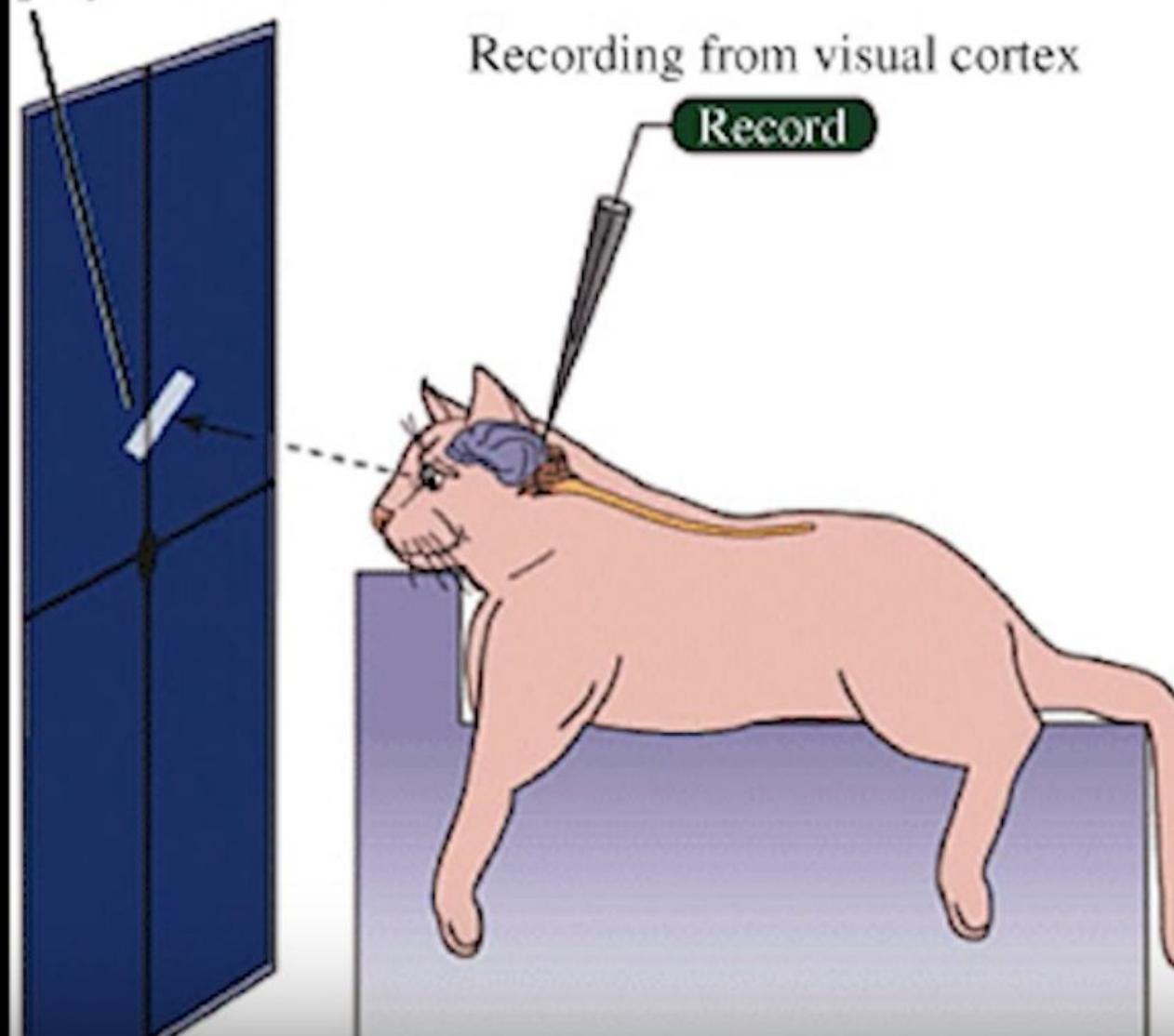


- (A) Cave painting at Chauvet, France, about 30,000 B.C.;
- (B) Aerial photograph of the picture of a monkey as part of the Nazca Lines geoglyphs, Peru, about 700 – 200 B.C.;
- (C) Shen Zhou (1427-1509 A.D.): Poet on a mountain top, ink on paper, China;
- (D) Line drawing by 7-year old I. Lleras (2010 A.D.).



### A Experimental setup

Light bar stimulus  
projected on screen



### B Stimulus orientation



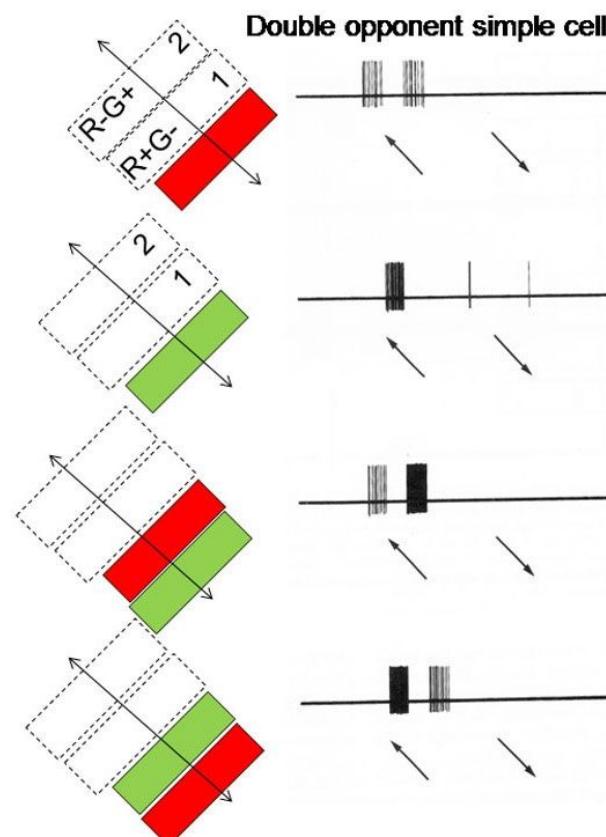
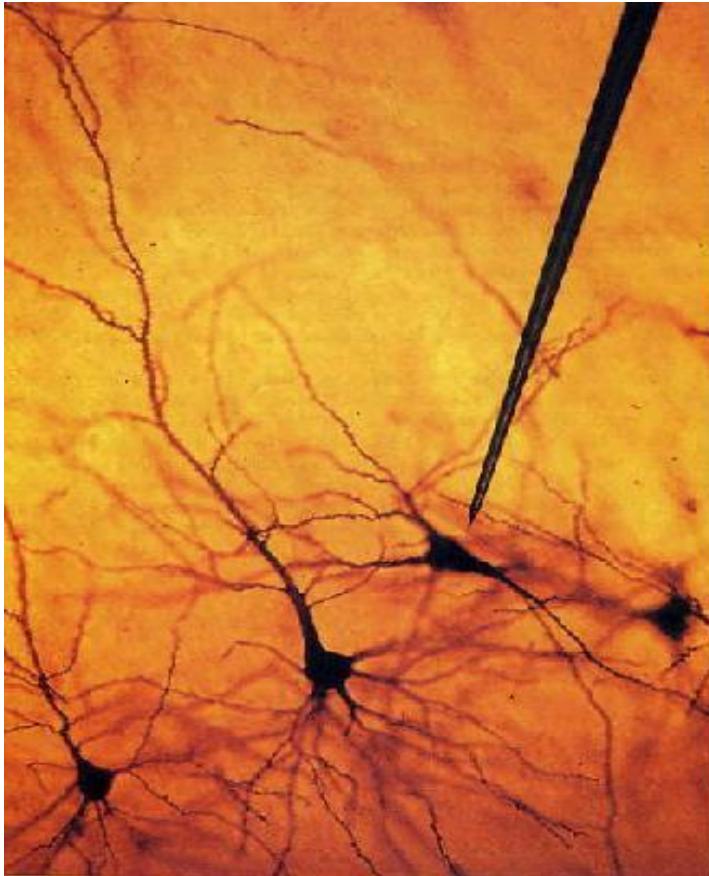
### Stimulus presented



Hubel & Wiesel, 1960s

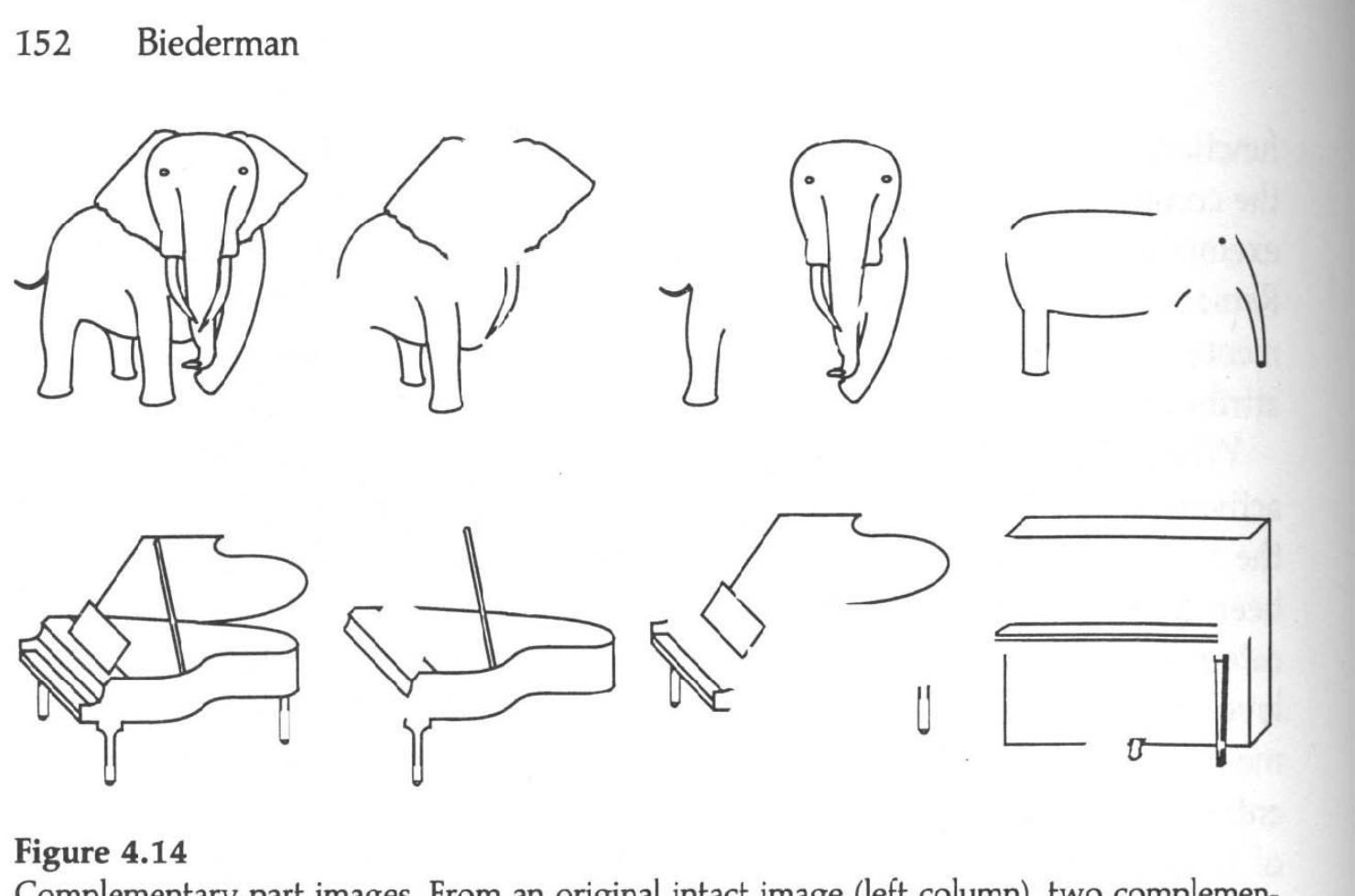
Jan 16, 2025

We know edges are special from human  
(mammalian) vision studies



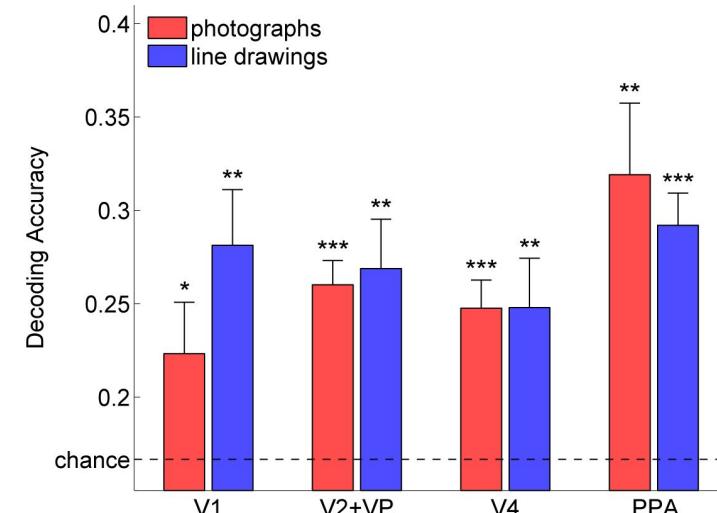
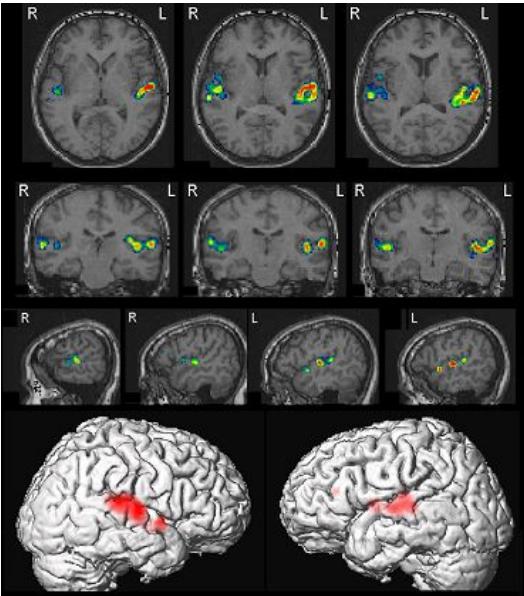
We know edges are special from human  
(mammalian) vision studies

152 Biederman



**Figure 4.14**

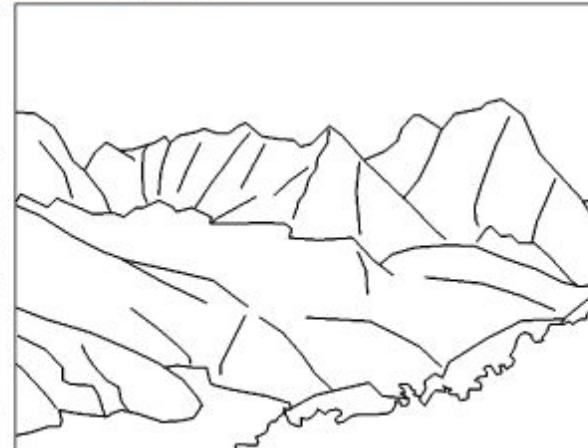
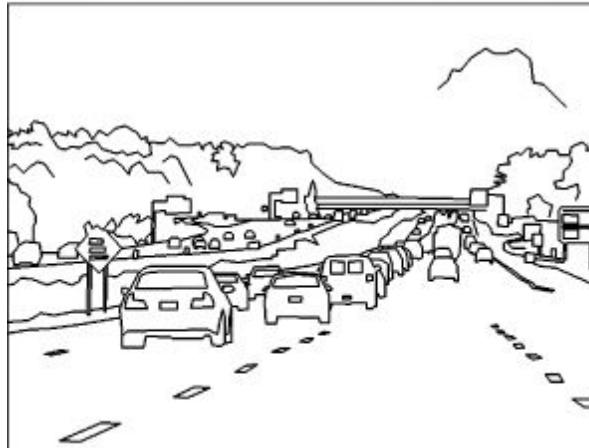
Complementary-part images. From an original intact image (left column), two complemen-



Walther, Chai, Caddigan, Beck & Fei-Fei, *PNAS*, 2011

# Edge detection

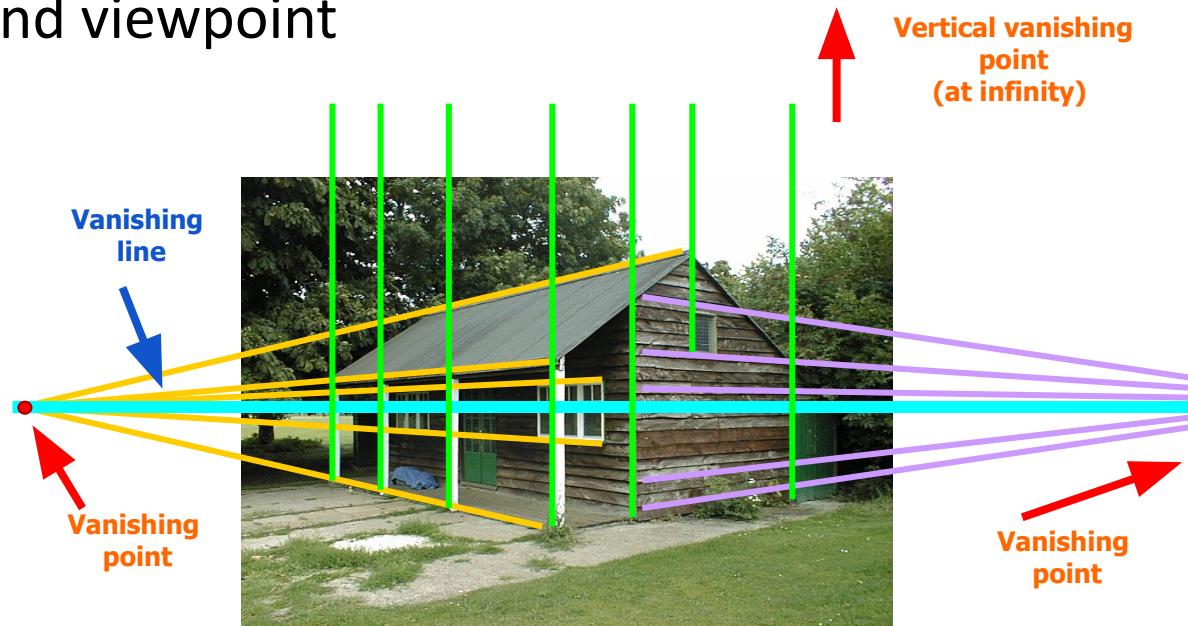
- **Goal:** Identify sudden changes (discontinuities) in an image
  - Intuitively, most semantic and shape information from the image can be encoded in the edges
  - More compact than pixels
- **Ideal:** artist's line drawing (but artist is also using object-level knowledge)



# Why do we care about edges?

- Extract information, recognize objects

- Recover geometry and viewpoint



# Origins of edges



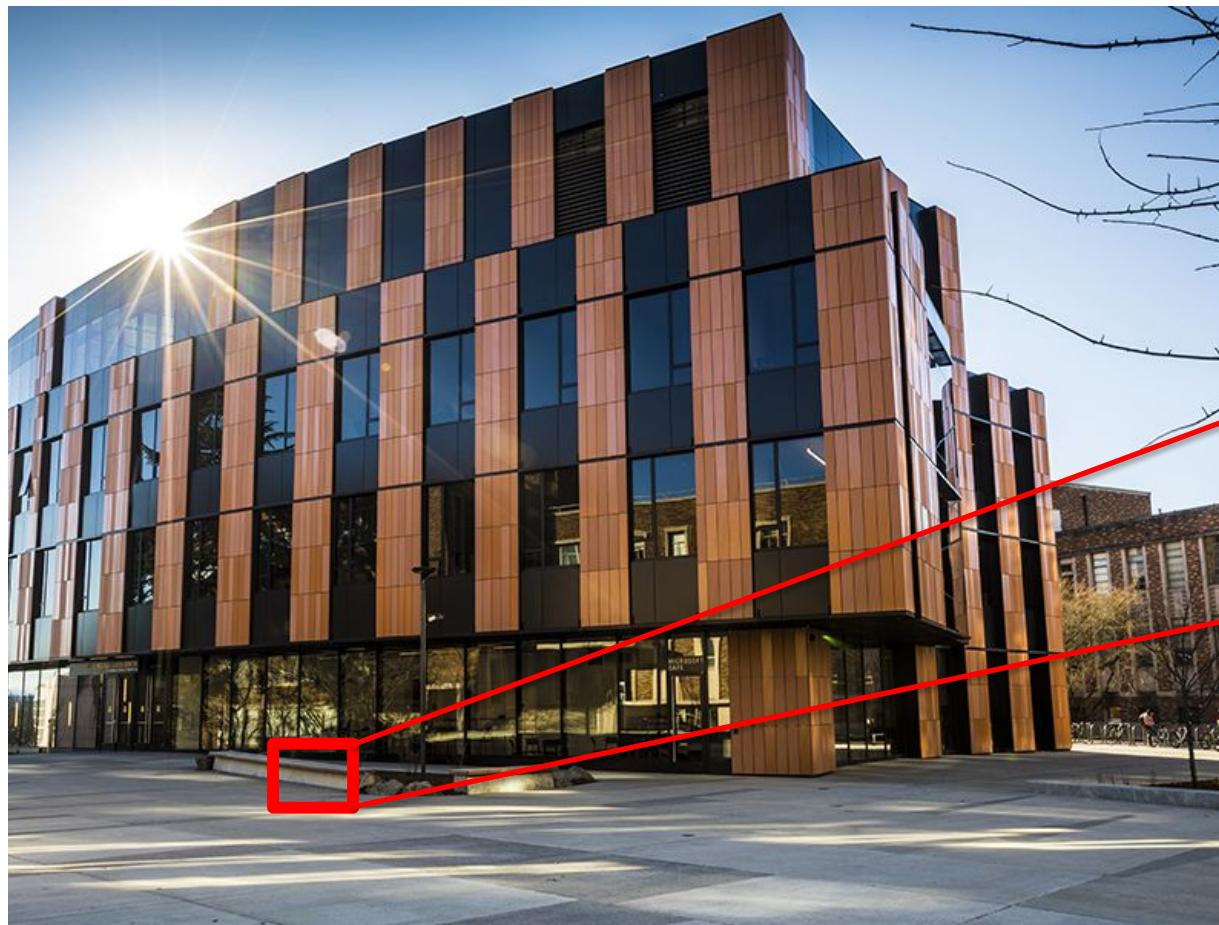
surface normal discontinuity

depth discontinuity

surface color discontinuity

illumination discontinuity

# Closeup of edges



Surface normal discontinuity



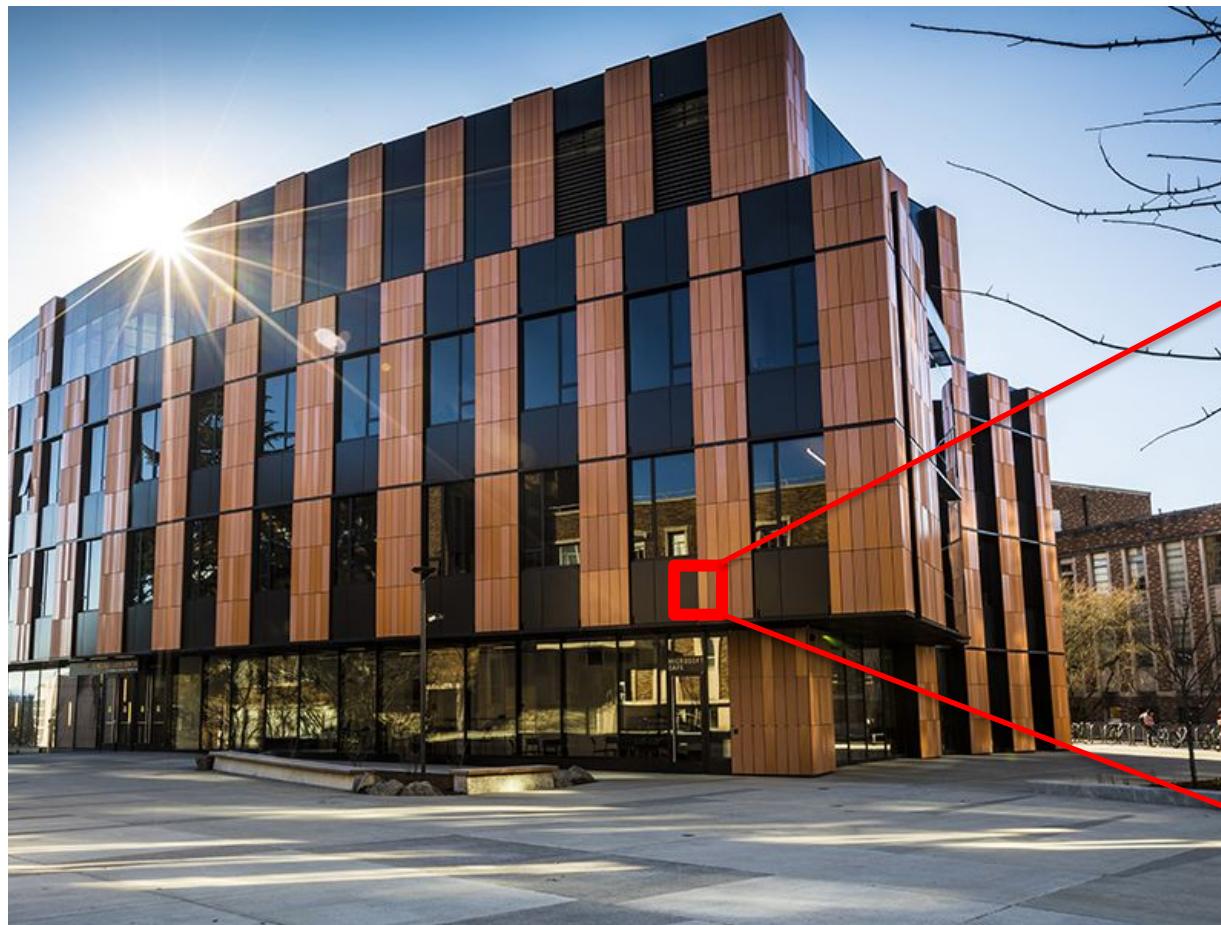
# Closeup of edges



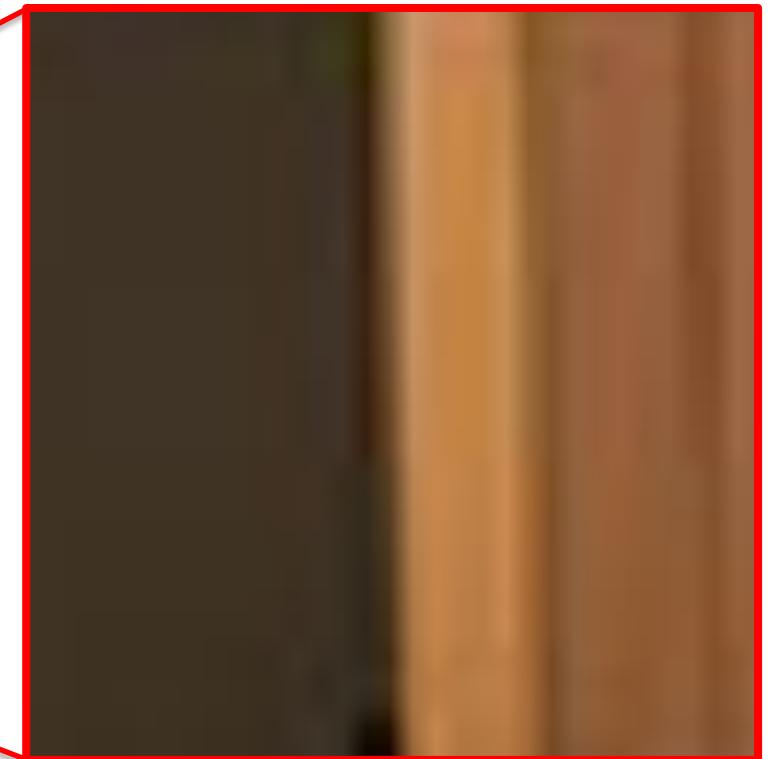
Depth discontinuity



# Closeup of edges



Surface color discontinuity



# What we will learn today

- Convolutions and Cross-Correlation
- Edge detection
- **Image Gradients**
- A simple edge detector

# Review: Derivatives in 1D - example

$$y = x^2 + x^4$$

Q. What is the  $dy/dx$ ?

# Review: Derivatives in 1D - example

$$y = x^2 + x^4$$

$$\frac{dy}{dx} = 2x + 4x^3$$

# Derivatives in 1D - example

$$y = x^2 + x^4$$

$$y = \sin x + e^{-x}$$

$$\frac{dy}{dx} = 2x + 4x^3$$

Q. What is the  $dy/dx$ ?

# Derivatives in 1D - example

$$y = x^2 + x^4$$

$$\frac{dy}{dx} = 2x + 4x^3$$

$$y = \sin x + e^{-x}$$

$$\frac{dy}{dx} = \cos x + (-1)e^{-x}$$

# Approximating derivatives using numerical differentiation

$$\frac{df}{dx} = \lim_{\Delta x=0} \frac{f[x + \Delta x] - f[x]}{\Delta x} = f'(x) = f_x$$

# Approximating derivatives using numerical differentiation

$$\frac{df}{dx} = \lim_{\Delta x=0} \frac{\text{Change in } f \text{ at } x}{\text{Change in } x} = \frac{f[x + \Delta x] - f[x]}{\Delta x} = f'(x) = f_x$$

In discrete derivatives with images, smallest value of x is 1 pixel

$$\begin{aligned}\frac{df}{dx} &= \lim_{\Delta x=0} \frac{f[x + \Delta x] - f[x]}{\Delta x} = f'(x) = f_x \\ &= \frac{f[x + 1] - f[x]}{1} \\ &= f[x + 1] - f[x]\end{aligned}$$

This is called a forward derivative

But change at  $x$  can be measured in many different ways

$$\frac{df}{dx} = f[x] - f[x - 1]$$

Backward

But change at  $x$  can be measured in many different ways

$$\begin{aligned}\frac{df}{dx} &= f[x] - f[x - 1] && \text{Backward} \\ &= f[x + 1] - f[x] && \text{Forward}\end{aligned}$$

But change at  $x$  can be measured in many different ways

$$\frac{df}{dx} = f[x] - f[x - 1] \quad \text{Backward}$$

$$= f[x + 1] - f[x] \quad \text{Forward}$$

$$= \frac{1}{2}(f[x + 1] - f[x - 1]) \quad \text{Central}$$

# Designing filters that perform differentiation

- Using Backward differentiation

$$g[n, m] = ??$$

Q. What is the equation in width (2nd) dimension?

# Designing filters that perform differentiation

- Using Backward differentiation

$$g[n, m] = f[n, m] - f[n, m - 1]$$

# Designing filters that perform differentiation

- Using Backward differentiation

$$g[n, m] = f[n, m] - f[n, m - 1]$$

Q. Let's write this as a filter

Remember the moving average filter:

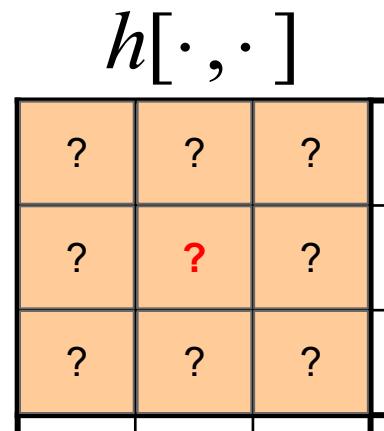
$$\frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

# Designing filters that perform differentiation

- Using Backward differentiation

$$g[n, m] = f[n, m] - f[n, m - 1]$$

Q. Let's write this as a filter



Remember the moving average filter:

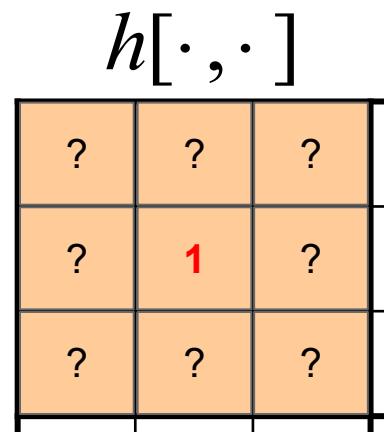
$$\frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

# Designing filters that perform differentiation

- Using Backward differentiation

$$g[n, m] = f[n, m] - f[n, m - 1]$$

Q. Let's write this as a filter



Remember the moving average filter:

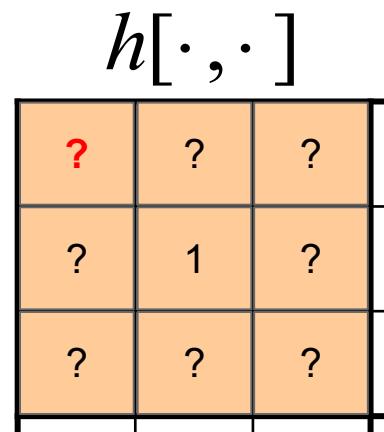
$$\frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

# Designing filters that perform differentiation

- Using Backward differentiation

$$g[n, m] = f[n, m] - f[n, m - 1]$$

Q. Let's write this as a filter



Remember the moving average filter:

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

# Designing filters that perform differentiation

- Using Backward differentiation

$$g[n, m] = f[n, m] - f[n, m - 1]$$

Q. Let's write this as a filter

$h[\cdot, \cdot]$

0	?	?
?	1	?
?	?	?

Remember the moving average filter:

$h[\cdot, \cdot]$

$\frac{1}{9}$	1	1	1
1	1	1	1
1	1	1	1

# Designing filters that perform differentiation

- Using Backward differentiation

$$g[n, m] = f[n, m] - f[n, m - 1]$$

Q. Let's write this as a filter

$h[\cdot, \cdot]$

0	?	?
?	1	?
?	?	?

Remember the moving average filter:

$h[\cdot, \cdot]$

$\frac{1}{9}$	1	1	1
1	1	1	1
1	1	1	1

# Designing filters that perform differentiation

- Using Backward differentiation

$$g[n, m] = f[n, m] - f[n, m - 1]$$

Q. Let's write this as a filter

$$h[\cdot, \cdot]$$

0	0	0
?	1	?
?	?	?

Remember the moving average filter:

$$h[\cdot, \cdot]$$

1	1	1
1	1	1
1	1	1

$$\frac{1}{9}$$

# Designing filters that perform differentiation

- Using Backward differentiation

$$g[n, m] = f[n, m] - f[n, m - 1]$$

Q. Let's write this as a filter

$h[\cdot, \cdot]$

0	0	0
?	1	?
?	?	?

Remember the moving average filter:

$h[\cdot, \cdot]$

$\frac{1}{9}$	1	1	1
1	1	1	1
1	1	1	1

# Designing filters that perform differentiation

- Using Backward differentiation

$$g[n, m] = f[n, m] - f[n, m - 1]$$

Q. Let's write this as a filter

$h[\cdot, \cdot]$

0	0	0
?	1	?
0	0	0

Remember the moving average filter:

$h[\cdot, \cdot]$

$\frac{1}{9}$	1	1	1
1	1	1	1
1	1	1	1

# Designing filters that perform differentiation

- Using Backward differentiation

$$g[n, m] = f[n, m] - f[n, m - 1]$$

Q. Last ones: What are these two?

$$h[\cdot, \cdot]$$

0	0	0
?	1	?
0	0	0

Remember the moving average filter:

$$h[\cdot, \cdot]$$

1	1	1
1	1	1
1	1	1

$$\frac{1}{9}$$

# Designing filters that perform differentiation

- Using Backward differentiation

$$g[n, m] = f[n, m] - f[n, m - 1]$$

Q. Last ones: What are these two?

$h[\cdot, \cdot]$

0	0	0
0	1	-1
0	0	0

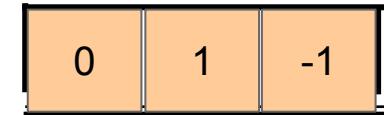
Remember the moving average filter:

$h[\cdot, \cdot]$

$\frac{1}{9}$	1	1	1
1	1	1	1
1	1	1	1

# Designing filters that perform differentiation

- Using Backward differentiation:



$$g[n, m] = f[n, m] - f[n, m - 1]$$

Remember the moving average filter:

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

# Designing filters that perform differentiation

- Using Backward differentiation:

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

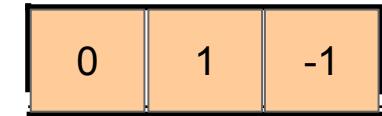
$$g[n, m] = f[n, m] - f[n, m - 1]$$

- Using Forward differentiation:

Q. What is the formula?

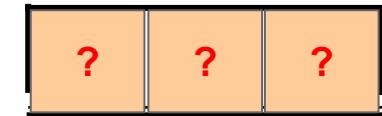
# Designing filters that perform differentiation

- Using Backward differentiation:



$$g[n, m] = f[n, m] - f[n, m - 1]$$

- Using Forward differentiation:

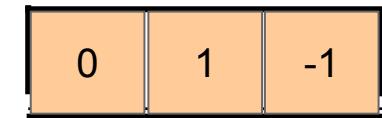


$$g[n, m] = f[n, m + 1] - f[n, m]$$

Q. What is the filter look like?

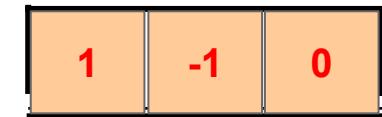
# Designing filters that perform differentiation

- Using Backward differentiation:



$$g[n, m] = f[n, m] - f[n, m - 1]$$

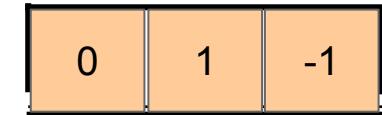
- Using Forward differentiation:



$$g[n, m] = f[n, m + 1] - f[n, m]$$

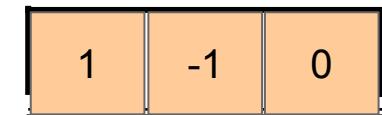
# Designing filters that perform differentiation

- Using Backward differentiation:



$$g[n, m] = f[n, m] - f[n, m - 1]$$

- Using Forward differentiation:



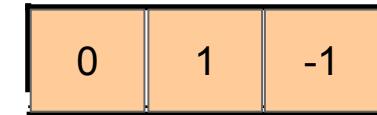
$$g[n, m] = f[n, m + 1] - f[n, m]$$

- Using Central differentiation:

Q. What is the formula?

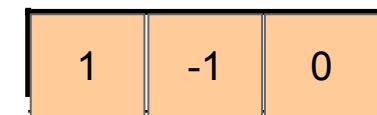
# Designing filters that perform differentiation

- Using Backward differentiation:



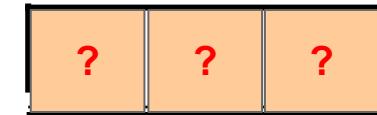
$$g[n, m] = f[n, m] - f[n, m - 1]$$

- Using Forward differentiation:



$$g[n, m] = f[n, m + 1] - f[n, m]$$

- Using Central differentiation:

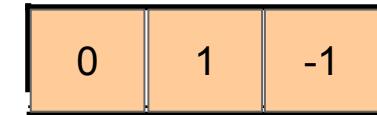


Q. What is  
the filter?

$$g[n, m] = f[n, m + 1] - f[n, m - 1]$$

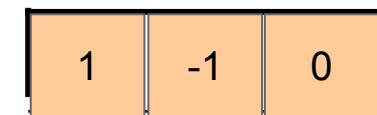
# Designing filters that perform differentiation

- Using Backward differentiation:



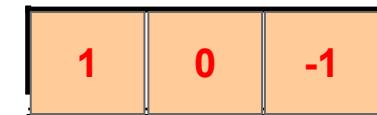
$$g[n, m] = f[n, m] - f[n, m - 1]$$

- Using Forward differentiation:



$$g[n, m] = f[n, m + 1] - f[n, m]$$

- Using Central differentiation:



$$g[n, m] = f[n, m + 1] - f[n, m - 1]$$

# Derivative in width dimension for one row

Using backward differentiation:

$$g[n, m] = f[n, m] - f[n, m - 1]$$

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

$$f[0, :] = [10, 15, 10, 10, 25, 20, 20, 20]$$

# Derivative in width dimension for one row

Using backward differentiation:

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

$$g[n, m] = f[n, m] - f[n, m - 1]$$

$$f[0, :] = [10, 15, 10, 10, 25, 20, 20, 20]$$

$$\frac{df}{dm}[0, :] = [ \textcolor{red}{?} ]$$

# Derivative in width dimension for one row

Using backward differentiation:

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

$$g[n, m] = f[n, m] - f[n, m - 1]$$

$$f[0, :] = [10, 15, 10, 10, 25, 20, 20, 20]$$

$$\frac{df}{dm}[0, :] = [10, \textcolor{red}{?}]$$

$$= 0 \times \begin{array}{|c|} \hline -1 \\ \hline \end{array} + 10 \times \begin{array}{|c|} \hline 1 \\ \hline \end{array} + 15 \times \begin{array}{|c|} \hline 0 \\ \hline \end{array}$$

# Derivative in width dimension for one row

Using backward differentiation:

$$g[n, m] = f[n, m] - f[n, m - 1]$$

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

$$f[0, :] = [10, 15, 10, 10, 25, 20, 20, 20]$$

$$\frac{df}{dm}[0, :] = [10, \quad 5, \quad ?]$$

$$= 10x \begin{bmatrix} -1 \end{bmatrix} + 15x \begin{bmatrix} 1 \end{bmatrix} + 10x \begin{bmatrix} 0 \end{bmatrix}$$

# Derivative in width dimension for one row

Using backward differentiation:

$$g[n, m] = f[n, m] - f[n, m - 1]$$

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

$$f[0, :] = [10, 15, 10, 10, 25, 20, 20, 20]$$

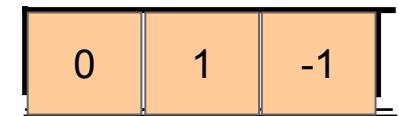
$$\frac{df}{dm}[0, :] = [10, 5, -5, ?]$$

$$= 15x \begin{matrix} -1 \end{matrix} + 10x \begin{matrix} 1 \end{matrix} + 10x \begin{matrix} 0 \end{matrix}$$

# Derivative in width dimension for one row

Using backward differentiation:

$$g[n, m] = f[n, m] - f[n, m - 1]$$



$$f[0, :] = [10, 15, 10, 10, 25, 20, 20, 20]$$

$$\frac{df}{dm}[0, :] = [10, 5, -5, 0, ?]$$

$$\rightarrow = 10x \begin{matrix} -1 \end{matrix} + 10x \begin{matrix} 1 \end{matrix} + 25x \begin{matrix} 0 \end{matrix}$$

# Derivative in width dimension for one row

Using backward differentiation:

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

$$g[n, m] = f[n, m] - f[n, m - 1]$$

$$f[0, :] = [10, 15, 10, 10, 25, 20, 20, 20]$$

$$\frac{df}{dm}[0, :] = [10, 5, -5, 0, 15, ?, ?, ?]$$

$$= 0 \times \boxed{-1} + 10 \times \boxed{1} + 15 \times \boxed{0}$$

# Derivative in width dimension for one row

Using backward differentiation:

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

$$g[n, m] = f[n, m] - f[n, m - 1]$$

$$f[0, :] = [10, 15, 10, 10, 25, 20, 20, 20]$$

$$\frac{df}{dm}[0, :] = [10, 5, -5, 0, 15, -5, 0, 0]$$

# Discrete derivation in 2D:

Given function  $f[n, m]$

$$\text{Gradient filter } \nabla f[n, m] = \begin{bmatrix} \frac{df}{dn} \\ \frac{df}{dm} \end{bmatrix} = \begin{bmatrix} f_n \\ f_m \end{bmatrix}$$

# Discrete derivation in 2D:

Given function  $f[n, m]$

$$\text{Gradient filter } \nabla f[n, m] = \begin{bmatrix} \frac{df}{dn} \\ \frac{df}{dm} \end{bmatrix} = \begin{bmatrix} f_n \\ f_m \end{bmatrix}$$

$$\text{Gradient magnitude } |\nabla f[n, m]| = \sqrt{f_n^2 + f_m^2}$$

# Discrete derivation in 2D:

Given function  $f[n, m]$

$$\text{Gradient filter } \nabla f[n, m] = \begin{bmatrix} \frac{df}{dn} \\ \frac{df}{dm} \end{bmatrix} = \begin{bmatrix} f_n \\ f_m \end{bmatrix}$$

$$\text{Gradient magnitude } |\nabla f[n, m]| = \sqrt{f_n^2 + f_m^2}$$

$$\text{Gradient direction } \theta = \tan^{-1}\left(\frac{f_m}{f_n}\right)$$

# 2D discrete derivative - example

$$f[n, m] = \begin{bmatrix} 10 & 10 & 20 & 20 & 20 \\ 10 & 10 & 20 & 20 & 20 \\ 10 & 10 & 20 & 20 & 20 \\ 10 & 10 & 20 & 20 & 20 \\ 10 & 10 & 20 & 20 & 20 \end{bmatrix}$$

# 2D discrete derivative - example

$$f[n, m] = \begin{bmatrix} 10 & 10 & 20 & 20 & 20 \\ 10 & 10 & 20 & 20 & 20 \\ 10 & 10 & 20 & 20 & 20 \\ 10 & 10 & 20 & 20 & 20 \\ 10 & 10 & 20 & 20 & 20 \end{bmatrix} \quad h[n, m] = \begin{bmatrix} 1 \\ 0 \\ -1 \end{bmatrix}$$
$$g[n, m] = \begin{bmatrix} ? & ? & ? & ? & ? \end{bmatrix}$$

# 2D discrete derivative - example

$$f[n, m] = \begin{bmatrix} 10 & 20 & 20 & 20 \\ 10 & 20 & 20 & 20 \\ 10 & 10 & 20 & 20 \\ 10 & 10 & 20 & 20 \\ 10 & 10 & 20 & 20 \end{bmatrix}$$
$$h[n, m] = \begin{bmatrix} 1 \\ 0 \\ -1 \end{bmatrix}$$
$$g[n, m] = \begin{bmatrix} 10 & 10 & 20 & 20 \end{bmatrix}$$

$f[n, m]$  is a 5x4 matrix of values. A 3x3 kernel  $h[n, m]$  is applied to the top-left element (10) of  $f$ . The result is calculated as follows:

$= 0 \times$   $\boxed{-1}$

$+ 10 \times$   $\boxed{0}$

$+ 10 \times$   $\boxed{1}$

The final result is  $\boxed{10}$ .

# 2D discrete derivative - example

$$f[n, m] = \begin{bmatrix} 10 & 10 & 20 & 20 & 20 \\ 10 & 10 & 20 & 20 & 20 \\ 10 & 10 & 20 & 20 & 20 \\ 10 & 10 & 20 & 20 & 20 \\ 10 & 10 & 20 & 20 & 20 \end{bmatrix} \quad h[n, m] = \begin{bmatrix} 1 \\ 0 \\ -1 \end{bmatrix}$$
$$g[n, m] = \begin{bmatrix} 10 & 10 & 20 & 20 & 20 \\ ? & ? & ? & ? & ? \end{bmatrix}$$

# 2D discrete derivative - example

$$f[n, m] = \begin{bmatrix} 10 & 10 & 20 & 20 & 20 \\ 10 & 10 & 20 & 20 & 20 \\ 10 & 10 & 20 & 20 & 20 \\ 10 & 10 & 20 & 20 & 20 \\ 10 & 10 & 20 & 20 & 20 \end{bmatrix}$$
$$h[n, m] = \begin{bmatrix} 1 \\ 0 \\ -1 \end{bmatrix}$$
$$g[n, m] = \begin{bmatrix} 10 & 10 & 20 & 20 & 20 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$f[n, m]$  is a 5x5 matrix of values. A 3x3 kernel  $h[n, m]$  is applied to it. The result is  $g[n, m]$ .

The kernel  $h[n, m]$  is defined as:

= 10x	-1
+ 10x	0
+ 10x	1

Red arrows show the convolution process:

- A red box highlights the top-left 3x3 block of  $f[n, m]$ .
- Red arrows point from the highlighted block to the corresponding values in the kernel  $h[n, m]$ .
- A red box highlights the value 0 in the resulting matrix  $g[n, m]$ , which is the result of the convolution.

# 2D discrete derivative - example

$$f[n, m] = \begin{bmatrix} 10 & 10 & 20 & 20 & 20 \\ 10 & 10 & 20 & 20 & 20 \\ 10 & 10 & 20 & 20 & 20 \\ 10 & 10 & 20 & 20 & 20 \\ 10 & 10 & 20 & 20 & 20 \end{bmatrix}$$
$$h[n, m] = \begin{bmatrix} 1 \\ 0 \\ -1 \end{bmatrix}$$
$$g[n, m] = \begin{bmatrix} 10 & 10 & 20 & 20 & 20 \\ 0 & 0 & 0 & 0 & 0 \\ ? & ? & ? & ? & ? \end{bmatrix}$$

# 2D discrete derivative - example

$$f[n, m] = \begin{bmatrix} 10 & 10 & 20 & 20 & 20 \\ 10 & 10 & 20 & 20 & 20 \\ 10 & 10 & 20 & 20 & 20 \\ 10 & 10 & 20 & 20 & 20 \\ 10 & 10 & 20 & 20 & 20 \end{bmatrix}$$
$$h[n, m] = \begin{bmatrix} 1 \\ 0 \\ -1 \end{bmatrix}$$
$$g[n, m] = \begin{bmatrix} 10 & 10 & 20 & 20 & 20 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$f[n, m]$  is a 5x5 matrix of values. A 3x3 kernel  $h[n, m]$  is applied to it. The result is  $g[n, m]$ .

The kernel  $h[n, m]$  is defined as:

= 10x	-1
+ 10x	0
+ 10x	1

Red arrows show the convolution process. The highlighted element in  $g[1, 1]$  is the result of the convolution of the top-left 3x3 window of  $f$  with the kernel  $h$ .

# 2D discrete derivative - example

$$f[n, m] = \begin{bmatrix} 10 & 10 & 20 & 20 & 20 \\ 10 & 10 & 20 & 20 & 20 \\ 10 & 10 & 20 & 20 & 20 \\ 10 & 10 & 20 & 20 & 20 \\ 10 & 10 & 20 & 20 & 20 \end{bmatrix}$$
$$h[n, m] = \begin{bmatrix} 1 \\ 0 \\ -1 \end{bmatrix}$$
$$g[n, m] = \begin{bmatrix} 10 & 10 & 20 & 20 & 20 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ ? & ? & ? & ? & ? \end{bmatrix}$$

# 2D discrete derivative - example

$$f[n, m] = \begin{bmatrix} 10 & 10 & 20 & 20 & 20 \\ 10 & 10 & 20 & 20 & 20 \\ 10 & 10 & 20 & 20 & 20 \\ 10 & 10 & 20 & 20 & 20 \\ 10 & 10 & 20 & 20 & 20 \end{bmatrix}$$
$$h[n, m] = \begin{bmatrix} 1 \\ 0 \\ -1 \end{bmatrix}$$
$$g[n, m] = \begin{bmatrix} 10 & 10 & 20 & 20 & 20 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$f[n, m]$  is a 5x5 matrix of values. A 3x3 kernel  $h[n, m]$  is applied to it. The result is  $g[n, m]$ . The kernel is defined by coefficients:

$= 10x$	$-1$
$+ 10x$	$0$
$+ 10x$	$1$

Red arrows show the convolution process. The highlighted element in  $g[1, 1]$  is also highlighted in the input matrix  $f$ .

# 2D discrete derivative - example

$$f[n, m] = \begin{bmatrix} 10 & 10 & 20 & 20 & 20 \\ 10 & 10 & 20 & 20 & 20 \\ 10 & 10 & 20 & 20 & 20 \\ 10 & 10 & 20 & 20 & 20 \\ 10 & 10 & 20 & 20 & 20 \end{bmatrix}$$
$$h[n, m] = \begin{bmatrix} 1 \\ 0 \\ -1 \end{bmatrix}$$
$$g[n, m] = \begin{bmatrix} 10 & 10 & 20 & 20 & 20 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ \textcolor{red}{?} & \textcolor{red}{?} & \textcolor{red}{?} & \textcolor{red}{?} & \textcolor{red}{?} \end{bmatrix}$$

# 2D discrete derivative - example

$$f[n, m] = \begin{bmatrix} 10 & 10 & 20 & 20 & 20 \\ 10 & 10 & 20 & 20 & 20 \\ 10 & 10 & 20 & 20 & 20 \\ 10 & 10 & 20 & 20 & 20 \\ 10 & 10 & 20 & 20 & 20 \end{bmatrix}$$
$$h[n, m] = \begin{bmatrix} 1 \\ 0 \\ -1 \end{bmatrix}$$
$$g[n, m] = \begin{bmatrix} 10 & 10 & 20 & 20 & 20 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ -10 & -10 & -20 & -20 & -20 \end{bmatrix}$$

Diagram illustrating the computation of the 2D discrete derivative. A 5x5 input matrix  $f[n, m]$  is shown, with its first row highlighted in red. A 3x3 kernel  $h[n, m]$  is applied to the first row of  $f$ . The result is a 5x5 output matrix  $g$ , where the first row is zeroed out by the kernel's stride. The value at index  $(1, 1)$  in  $g$  is highlighted in red, indicating it is the result of the convolution step.

The input matrix  $f$  is defined as follows:

= 10x	-1
+ 10x	0
+ 0x	1

The output matrix  $g$  is defined as follows:

10	10	20	20	20
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0
-10	-10	-20	-20	-20

# Let's do the other one

$$f[n, m] = \begin{bmatrix} 10 & 10 & 20 & 20 & 20 \\ 10 & 10 & 20 & 20 & 20 \\ 10 & 10 & 20 & 20 & 20 \\ 10 & 10 & 20 & 20 & 20 \\ 10 & 10 & 20 & 20 & 20 \end{bmatrix} \quad h[n, m] = [1 \quad 0 \quad -1]$$

# Let's do the other one

$$f[n, m] = \begin{bmatrix} 10 & 10 & 20 & 20 & 20 \\ 10 & 10 & 20 & 20 & 20 \\ 10 & 10 & 20 & 20 & 20 \\ 10 & 10 & 20 & 20 & 20 \\ 10 & 10 & 20 & 20 & 20 \end{bmatrix} \quad h[n, m] = [1 \quad 0 \quad -1]$$
$$g[n, m] = \begin{bmatrix} ? \\ ? \\ ? \\ ? \\ ? \end{bmatrix}$$

# Let's do the other one

$$f[n, m] = \begin{bmatrix} 10 & 10 & 20 & 20 & 20 \\ 10 & 10 & 20 & 20 & 20 \\ 10 & 10 & 20 & 20 & 20 \\ 10 & 10 & 20 & 20 & 20 \\ 10 & 10 & 20 & 20 & 20 \end{bmatrix}$$

A red circle highlights the top-left element '10' in the first row and second column of the matrix.

$$h[n, m] = [1 \quad 0 \quad -1]$$
$$= 0x \boxed{-1} + 10x \boxed{0} + 10x \boxed{1}$$

A red bracket groups the terms:  $= 0x \boxed{-1} + 10x \boxed{0} + 10x \boxed{1}$ . A red arrow points from this bracket to the highlighted '10' in the matrix.

$$g[n, m] = \begin{bmatrix} 10 \\ 10 \\ 10 \\ 10 \\ 10 \end{bmatrix}$$

A red arrow points from the highlighted '10' in the matrix to the first element '10' in the vector  $g[n, m]$ .

# Let's do the other one

$$f[n, m] = \begin{bmatrix} 10 & 10 & 20 & 20 & 20 \\ 10 & 10 & 20 & 20 & 20 \\ 10 & 10 & 20 & 20 & 20 \\ 10 & 10 & 20 & 20 & 20 \\ 10 & 10 & 20 & 20 & 20 \end{bmatrix} \quad h[n, m] = [1 \quad 0 \quad -1]$$
$$g[n, m] = \begin{bmatrix} 10 & ? \\ 10 & ? \\ 10 & ? \\ 10 & ? \\ 10 & ? \end{bmatrix}$$

# Let's do the other one

$$f[n, m] = \begin{bmatrix} 10 & 10 & 20 & 20 & 20 \\ 10 & 10 & 20 & 20 & 20 \\ 10 & 10 & 20 & 20 & 20 \\ 10 & 10 & 20 & 20 & 20 \\ 10 & 10 & 20 & 20 & 20 \end{bmatrix}$$
$$h[n, m] = [1 \quad 0 \quad -1]$$
$$= 10x \boxed{-1} + 10x \boxed{0} + 20x \boxed{1}$$
$$g[n, m] = \begin{bmatrix} 10 & \boxed{10} \\ 10 & 10 \\ 10 & 10 \\ 10 & 10 \\ 10 & 10 \end{bmatrix}$$

# Let's do the other one

$$f[n, m] = \begin{bmatrix} 10 & 10 & 20 & 20 & 20 \\ 10 & 10 & 20 & 20 & 20 \\ 10 & 10 & 20 & 20 & 20 \\ 10 & 10 & 20 & 20 & 20 \\ 10 & 10 & 20 & 20 & 20 \end{bmatrix} \quad h[n, m] = [1 \quad 0 \quad -1]$$
$$g[n, m] = \begin{bmatrix} 10 & 10 & ? \\ 10 & 10 & ? \\ 10 & 10 & ? \\ 10 & 10 & ? \\ 10 & 10 & ? \end{bmatrix}$$

# Let's do the other one

$$f[n, m] = \begin{bmatrix} 10 & 10 & 20 & 20 & 20 \\ 10 & 10 & 20 & 20 & 20 \\ 10 & 10 & 20 & 20 & 20 \\ 10 & 10 & 20 & 20 & 20 \\ 10 & 10 & 20 & 20 & 20 \end{bmatrix}$$
$$h[n, m] = [1 \quad 0 \quad -1]$$
$$= 10x \quad -1 \quad + 20x \quad 0 \quad + 20x \quad 1$$
$$g[n, m] = \begin{bmatrix} 10 & 10 & 10 \\ 10 & 10 & 10 \\ 10 & 10 & 10 \\ 10 & 10 & 10 \end{bmatrix}$$

# Let's do the other one

$$f[n, m] = \begin{bmatrix} 10 & 10 & 20 & 20 & 20 \\ 10 & 10 & 20 & 20 & 20 \\ 10 & 10 & 20 & 20 & 20 \\ 10 & 10 & 20 & 20 & 20 \\ 10 & 10 & 20 & 20 & 20 \end{bmatrix} \quad h[n, m] = [1 \quad 0 \quad -1]$$
$$g[n, m] = \begin{bmatrix} 10 & 10 & 10 & ? \\ 10 & 10 & 10 & ? \\ 10 & 10 & 10 & ? \\ 10 & 10 & 10 & ? \\ 10 & 10 & 10 & ? \end{bmatrix}$$

# Let's do the other one

$$f[n, m] = \begin{bmatrix} 10 & 10 & 20 & 20 & 20 \\ 10 & 10 & 20 & 20 & 20 \\ 10 & 10 & 20 & 20 & 20 \\ 10 & 10 & 20 & 20 & 20 \\ 10 & 10 & 20 & 20 & 20 \end{bmatrix}$$
$$h[n, m] = [1 \quad 0 \quad -1]$$
$$= 20x \boxed{-1} + 20x \boxed{0} + 20x \boxed{1}$$
$$g[n, m] = \begin{bmatrix} 10 & 10 & 10 & \xrightarrow{0} \\ 10 & 10 & 10 & 0 \\ 10 & 10 & 10 & 0 \\ 10 & 10 & 10 & 0 \\ 10 & 10 & 10 & 0 \end{bmatrix}$$

# Let's do the other one

$$f[n, m] = \begin{bmatrix} 10 & 10 & 20 & 20 & 20 \\ 10 & 10 & 20 & 20 & 20 \\ 10 & 10 & 20 & 20 & 20 \\ 10 & 10 & 20 & 20 & 20 \\ 10 & 10 & 20 & 20 & 20 \end{bmatrix} \quad h[n, m] = [1 \quad 0 \quad -1]$$
$$g[n, m] = \begin{bmatrix} 10 & 10 & 10 & 0 & \textcolor{red}{?} \\ 10 & 10 & 10 & 0 & \textcolor{red}{?} \\ 10 & 10 & 10 & 0 & \textcolor{red}{?} \\ 10 & 10 & 10 & 0 & \textcolor{red}{?} \\ 10 & 10 & 10 & 0 & \textcolor{red}{?} \end{bmatrix}$$

# Let's do the other one

$$f[n, m] = \begin{bmatrix} 10 & 10 & 20 & 20 & 20 \\ 10 & 10 & 20 & 20 & 20 \\ 10 & 10 & 20 & 20 & 20 \\ 10 & 10 & 20 & 20 & 20 \\ 10 & 10 & 20 & 20 & 20 \end{bmatrix}$$
$$h[n, m] = [1 \quad 0 \quad -1]$$
$$= 20x \boxed{-1} + 20x \boxed{0} + 0x \boxed{1}$$
$$g[n, m] = \begin{bmatrix} 10 & 10 & 10 & 0 & \boxed{-20} \\ 10 & 10 & 10 & 0 & -20 \\ 10 & 10 & 10 & 0 & -20 \\ 10 & 10 & 10 & 0 & -20 \\ 10 & 10 & 10 & 0 & -20 \end{bmatrix}$$

The diagram illustrates the convolution process. A red arrow points from the highlighted value 20 in the input matrix to the output matrix. Another red arrow points from the highlighted value -1 in the kernel matrix to the output matrix. A third red arrow points from the highlighted value -20 in the output matrix to the right.

# 2D discrete derivative filters

Q. What does this filter do?

$$h[n, m] = \begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix}$$

# 2D discrete derivative filters

$$h[n, m] = \begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix}$$

Q. What does this filter do?

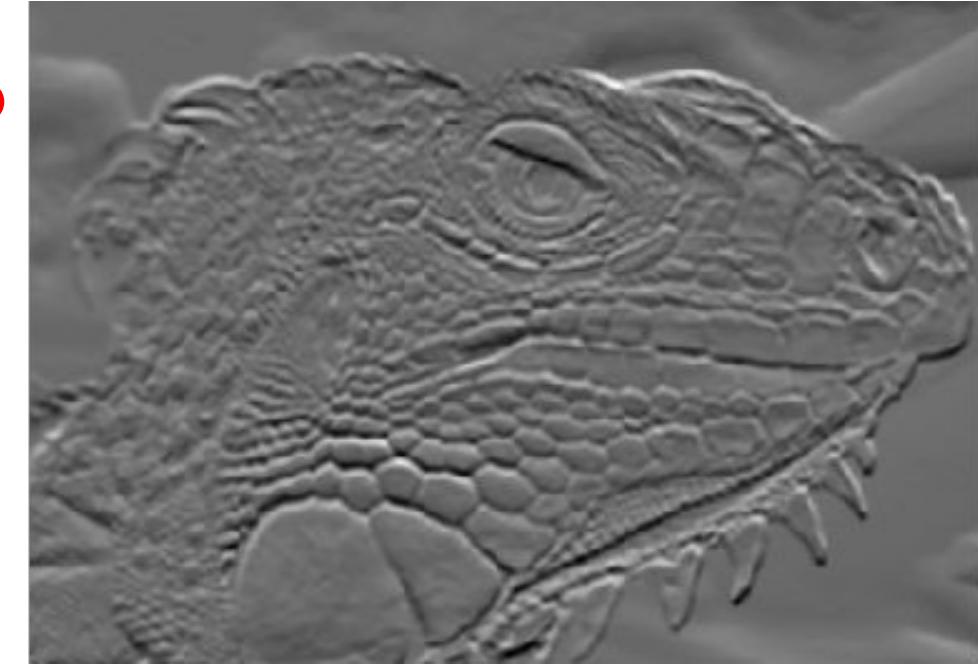
$$h[n, m] = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix}$$

Q. Which filter was applied?

$$\begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix} \quad \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix}$$

A

B



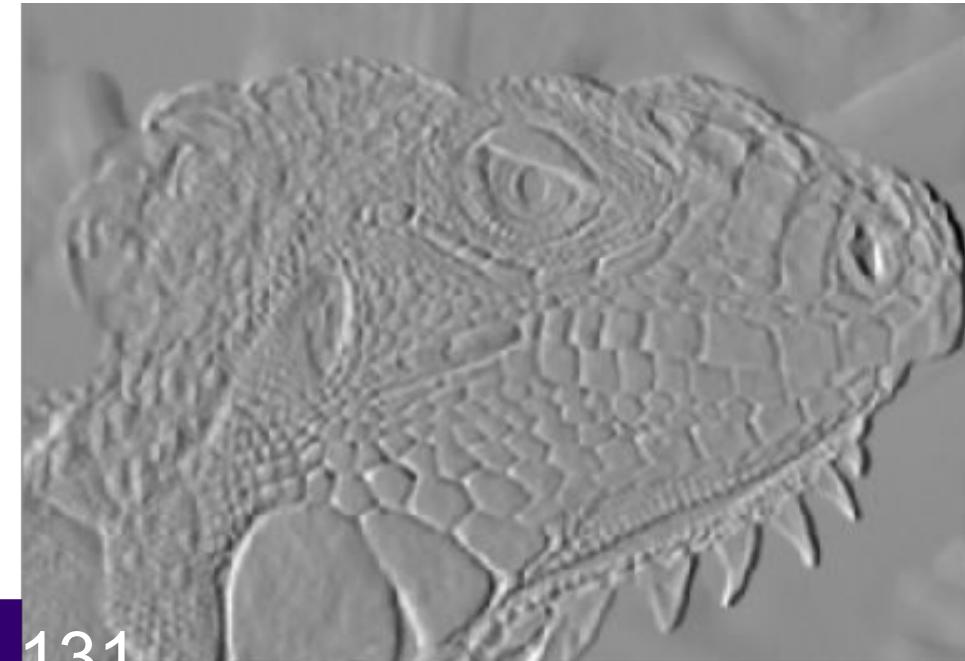
Q. Which filter was applied?

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

A

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

B

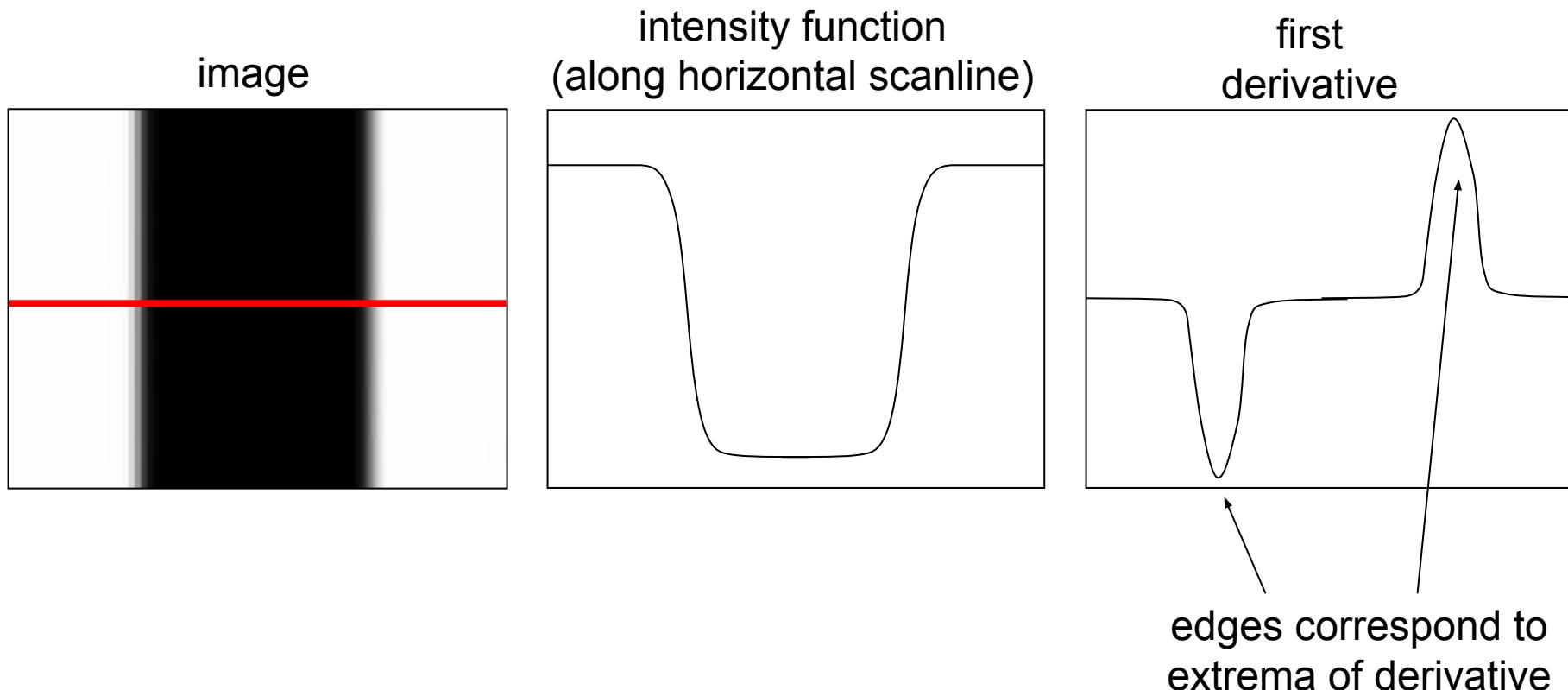


# What we will learn today

- Convolutions and Cross-Correlation
- Edge detection
- Image Gradients
- A simple edge detector

# Characterizing edges

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

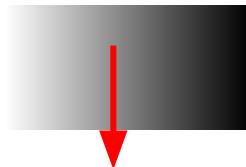


# Image gradient

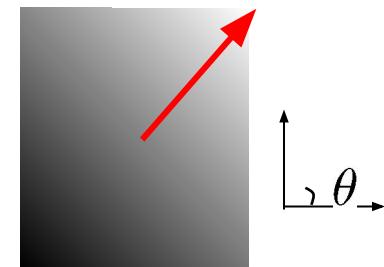
The gradient of an image:



$$\nabla_m f[n, m] = \begin{bmatrix} 0 & \frac{df}{dm} \end{bmatrix}$$



$$\nabla_n f[n, m] = \begin{bmatrix} \frac{df}{dn} & 0 \end{bmatrix}$$



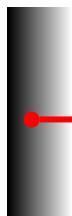
$$\nabla f[n, m] = \begin{bmatrix} \frac{df}{dn} & \frac{df}{dm} \end{bmatrix}$$

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

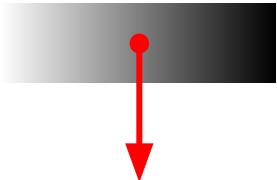
$$\theta = \tan^{-1}\left(\frac{f_m}{f_n}\right)$$

# Image gradient

The gradient of an image:

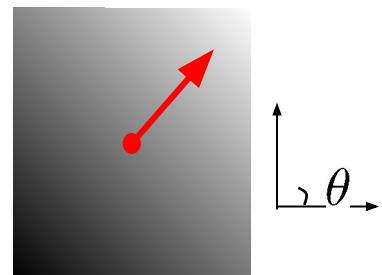


$$\nabla_m f[n, m] = \begin{bmatrix} 0 & \frac{df}{dm} \end{bmatrix}$$



$$\nabla_n f[n, m] = \begin{bmatrix} \frac{df}{dn} & 0 \end{bmatrix}$$

$$\nabla f[n, m] = \begin{bmatrix} \frac{df}{dn} & \frac{df}{dm} \end{bmatrix}$$



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

The *edge strength* is given by the gradient magnitude

$$|\nabla f[n, m]| = \sqrt{f_n^2 + f_m^2}$$

$$\theta = \tan^{-1}\left(\frac{f_m}{f_n}\right)$$

# Finite differences: example

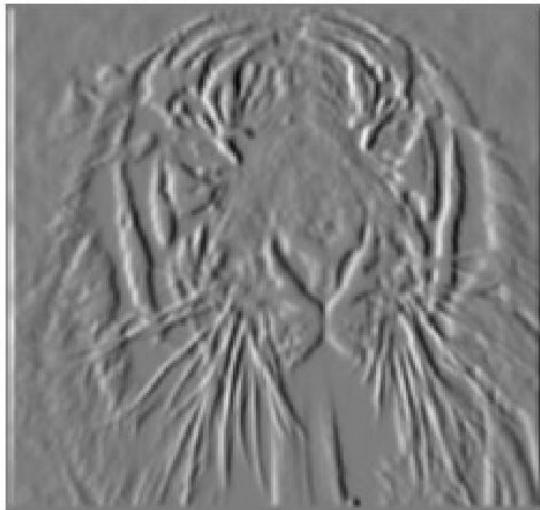
Original  
Image



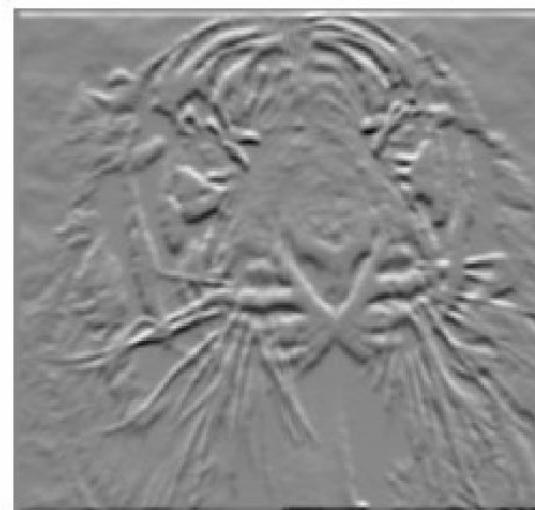
Gradient  
magnitude



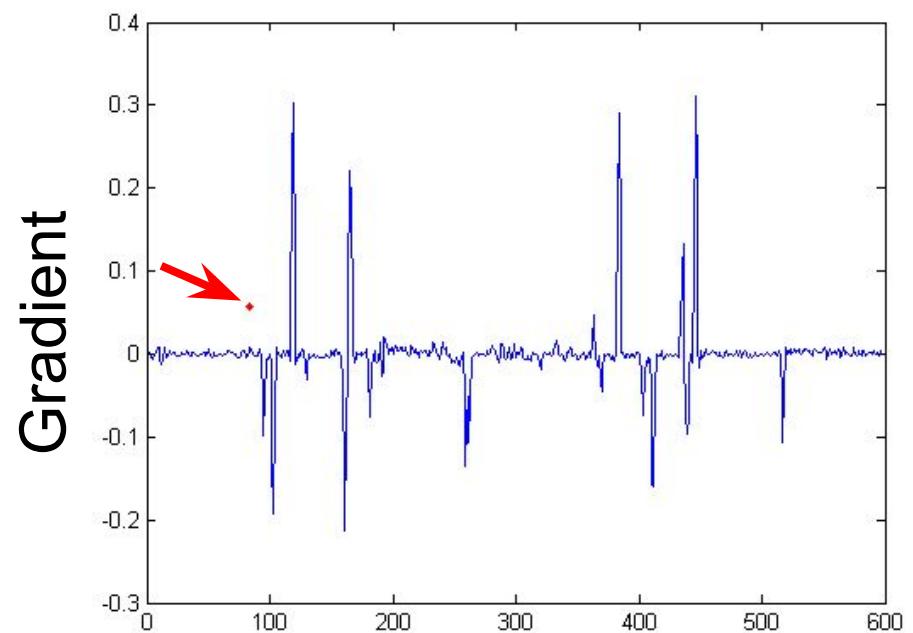
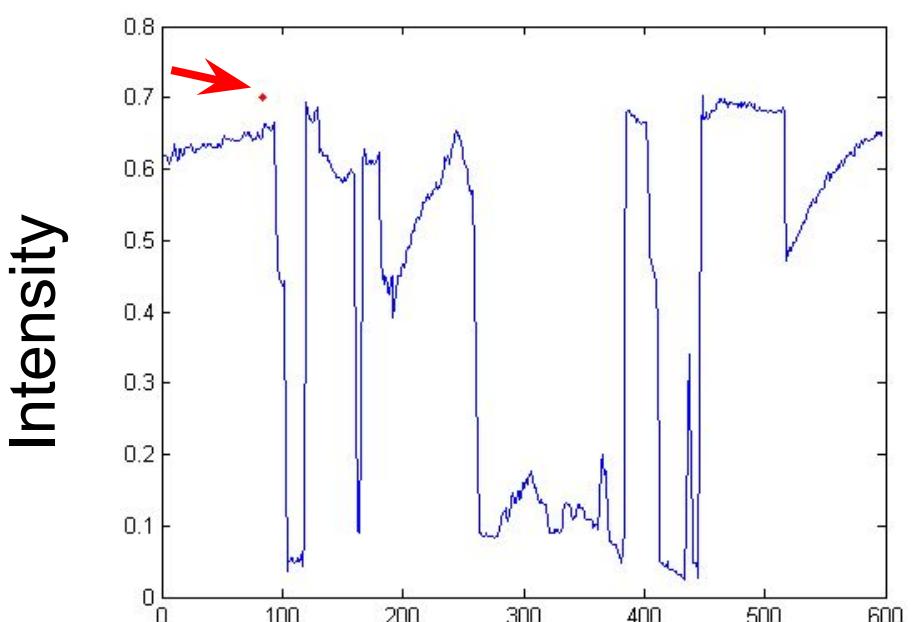
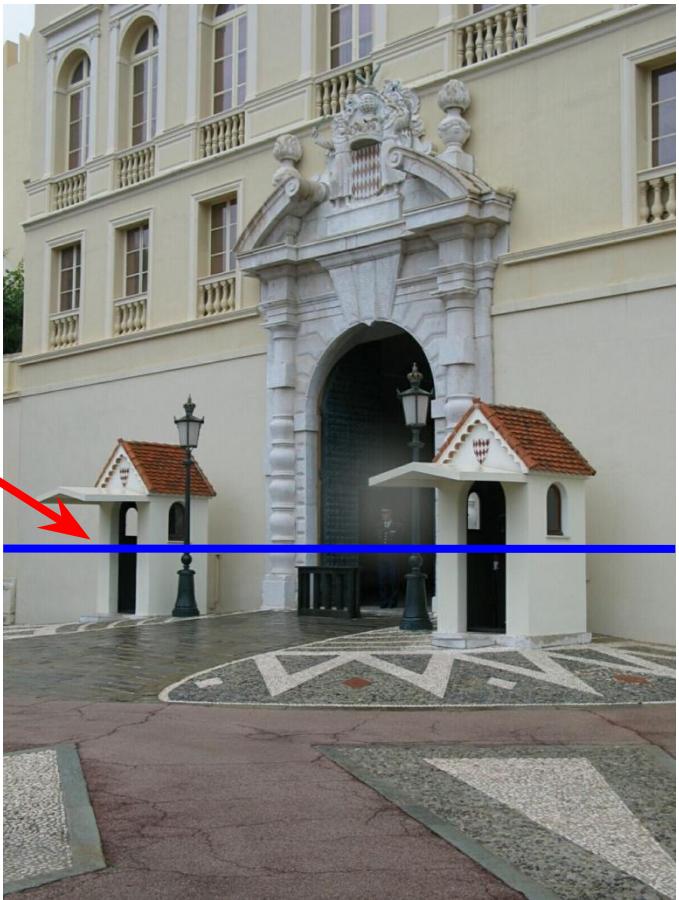
width-direction



height-direction



# Intensity profile



# Summary

- Convolutions and Cross-Correlation
- Edge detection
- Image Gradients
- A simple edge detector

# Next time: Detecting lines