

Lecture 5

Derivatives and edges

Administrative

A1 is out

- It is graded
- Due **Tue, Apr 16**

A2 will be out this weekend

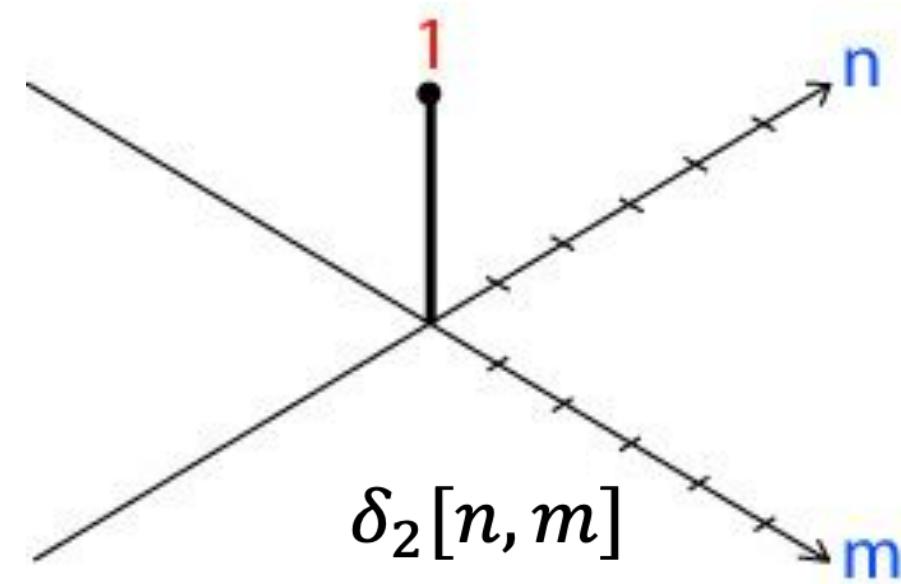
Administrative

Recitation this friday:
More linear algebra recap

by Mahtab

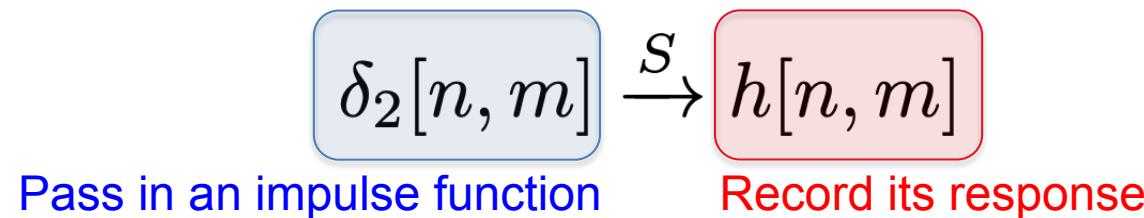
So far: 2D impulse function

- 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

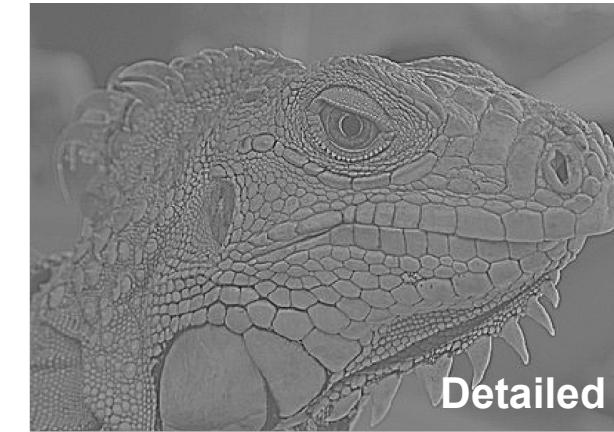


-



smoothed (3x3)

=

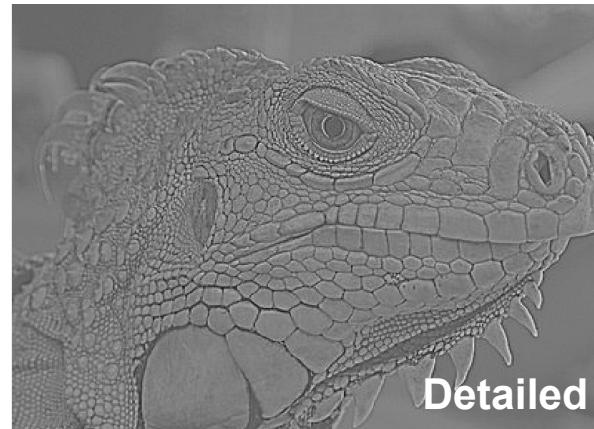


Detailed

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

What we missed last time: 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! $h_1 \ast\ast h_2 = h_2 \ast\ast h_1$

What we will learn today

- Edge detection
- Image Gradients
- A simple edge detector
- Sobel edge detector

Some background reading:

Forsyth and Ponce, Computer Vision, Chapter 8

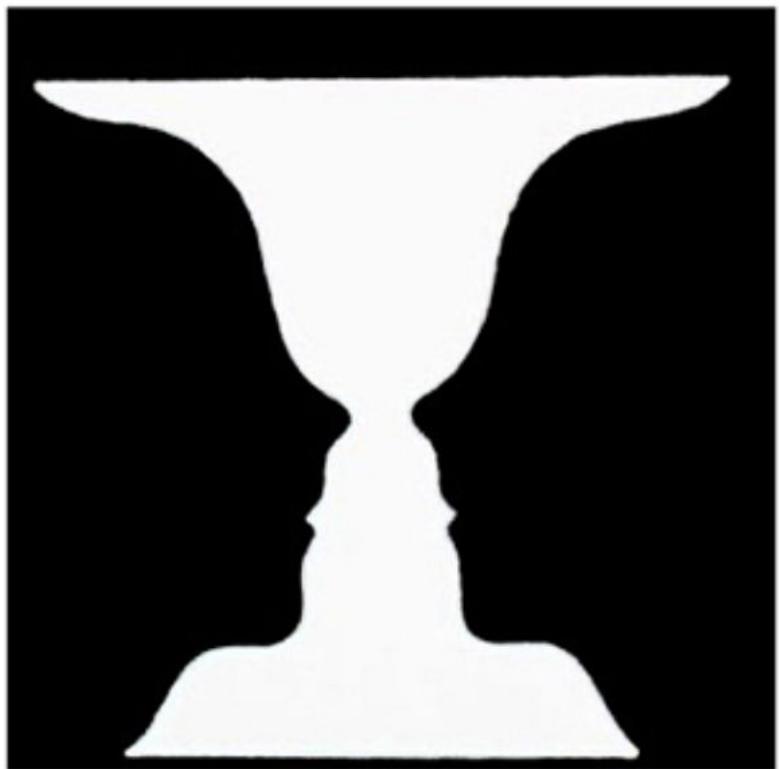
What we will learn today

- Edge detection
 - Image Gradients
 - A simple edge detector
 - Sobel edge detector

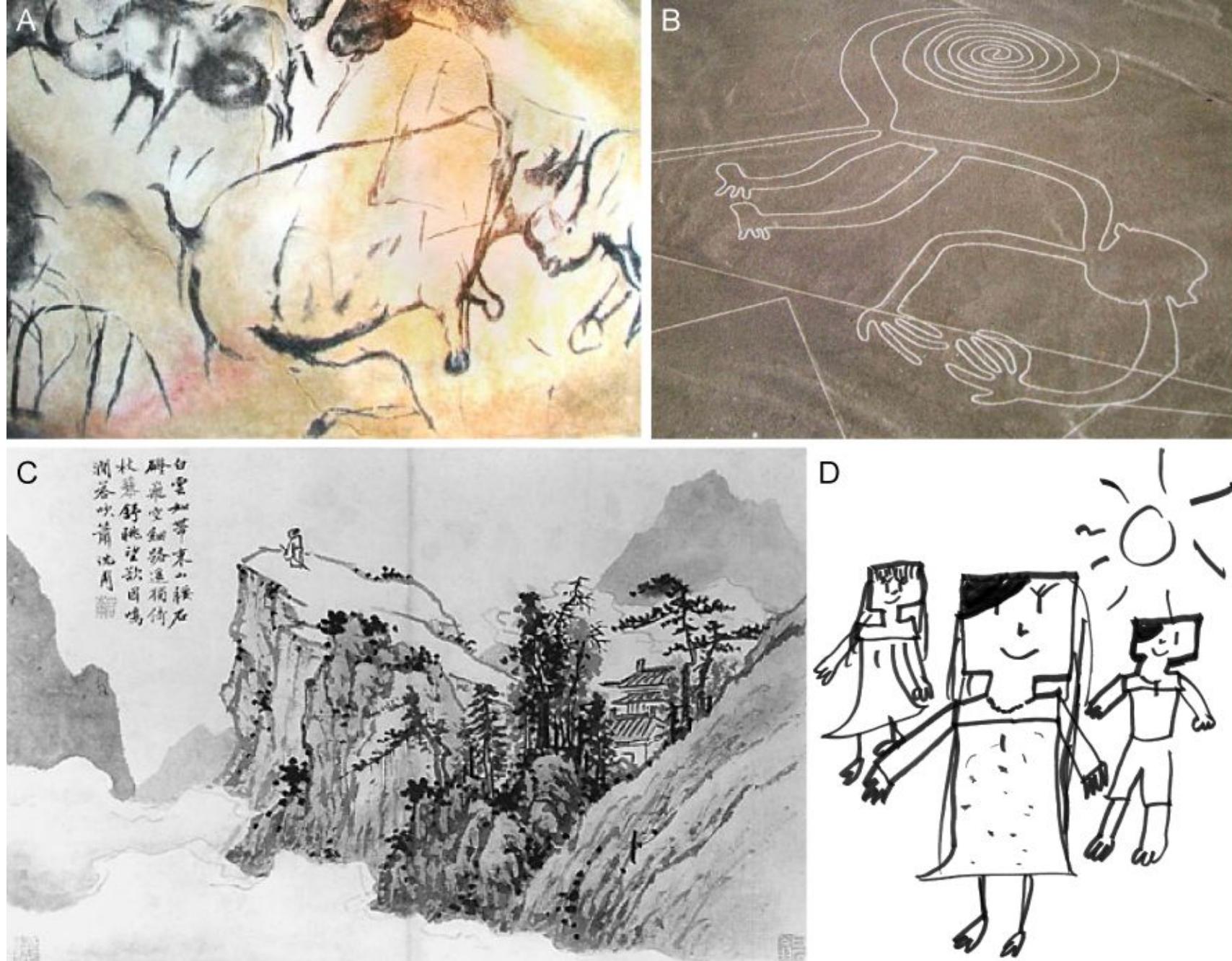
Some background reading:

Forsyth and Ponce, Computer Vision, Chapter 8

Q. What do you see?

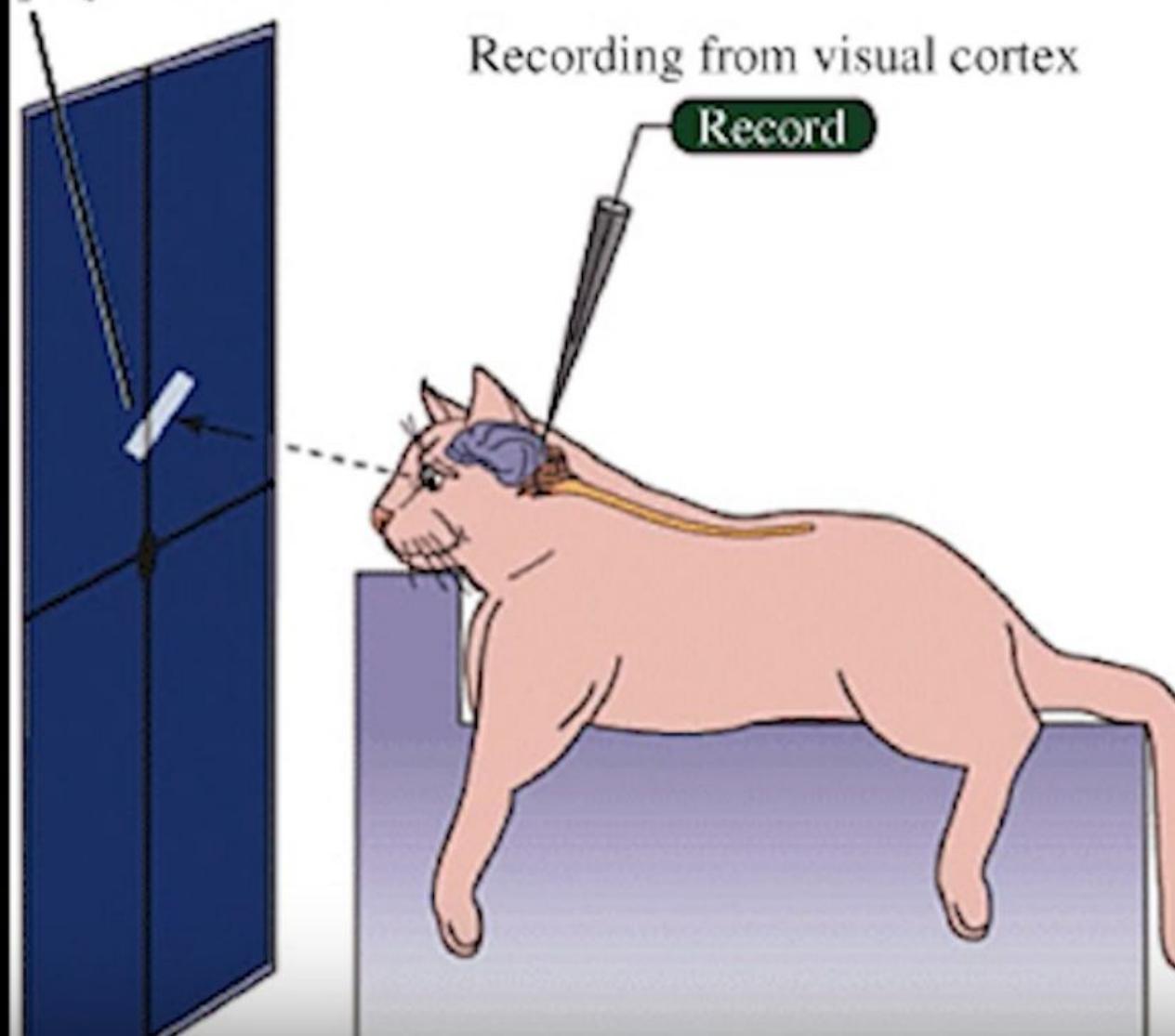


- (A) Cave painting at Chauvet, France, about 30,000 B.C.;
- (B) Aerial photograph of the picture of a monkey as part of the Nazca Lines geoglyphs, Peru, about 700 – 200 B.C.;
- (C) Shen Zhou (1427-1509 A.D.): Poet on a mountain top, ink on paper, China;
- (D) Line drawing by 7-year old I. Lleras (2010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



A Experimental setup

Light bar stimulus
projected on screen



B Stimulus orientation

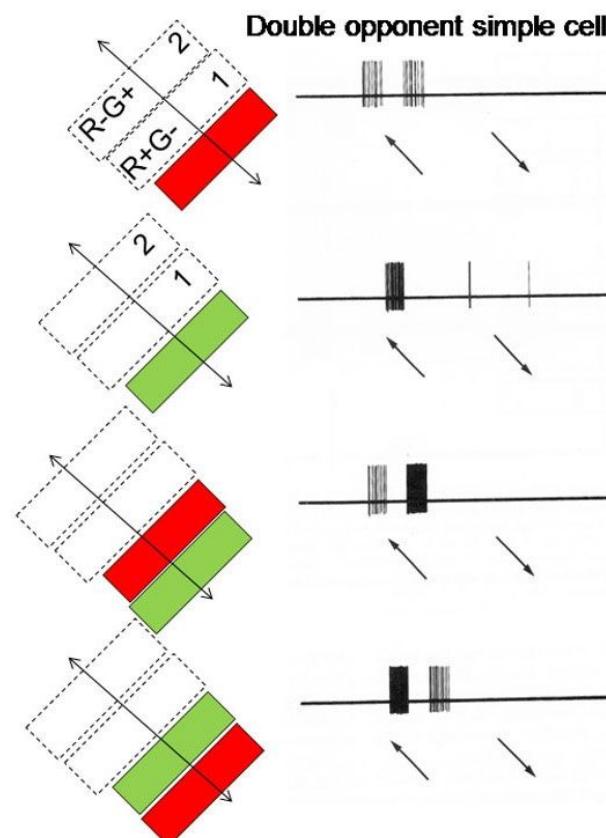
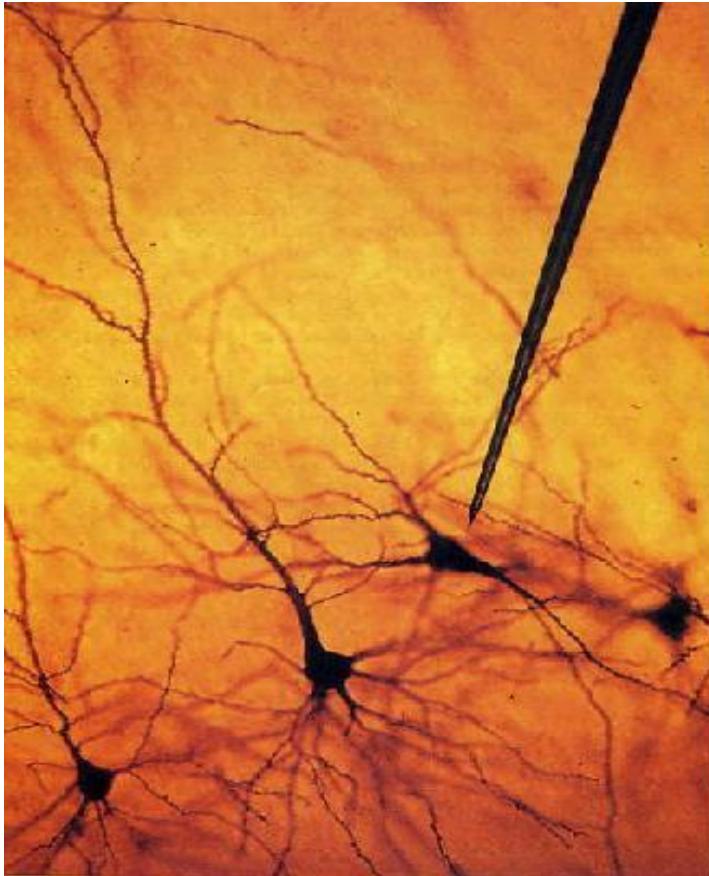


Stimulus presented



Hubel & Wiesel, 1960s

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



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

152 Biederman

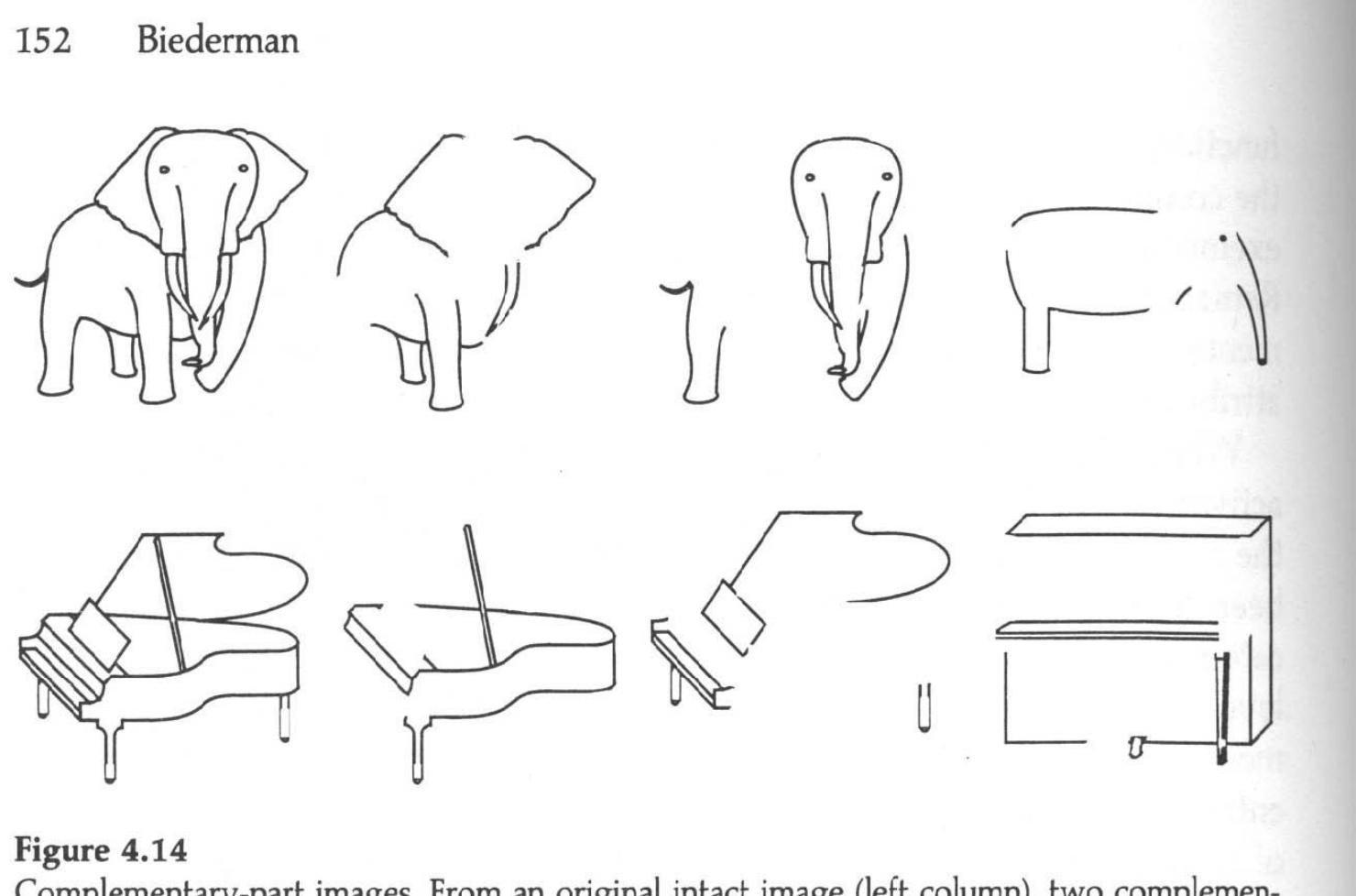
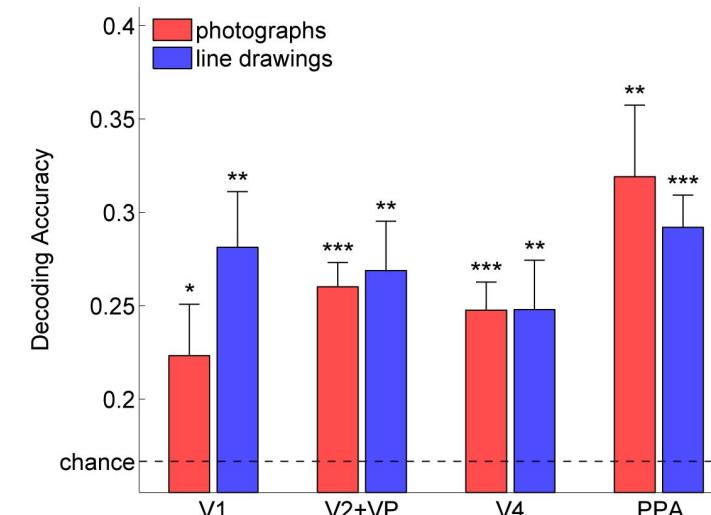
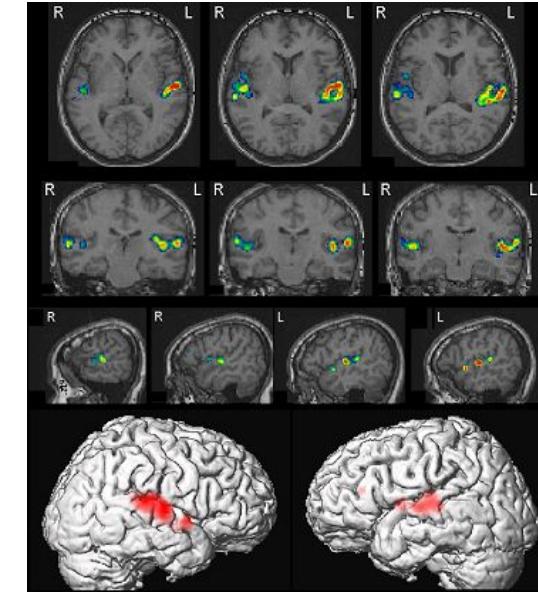


Figure 4.14

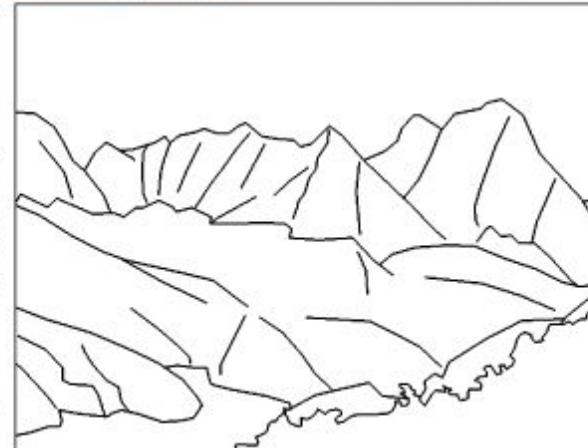
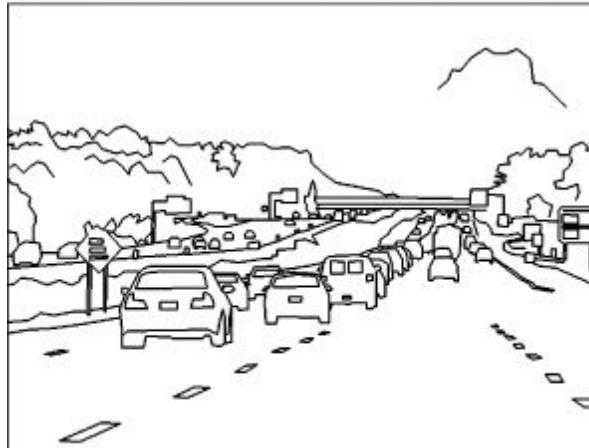
Complementary-part images. From an original intact image (left column), two complemen-



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

Edge detection

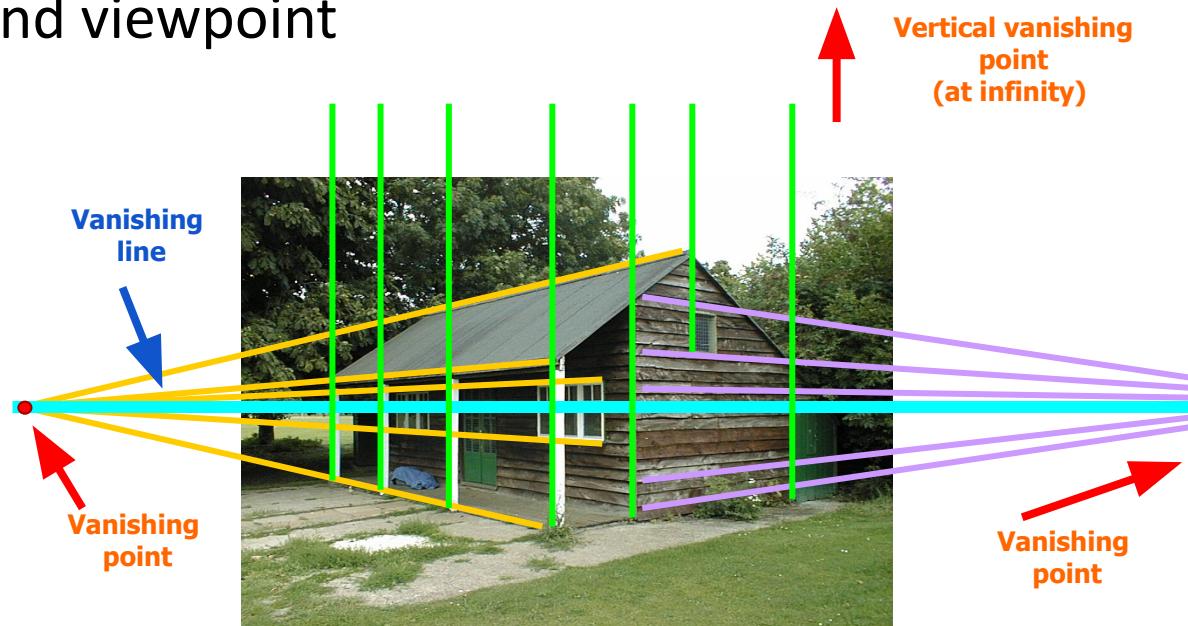
- **Goal:** Identify sudden changes (discontinuities) in an image
 - Intuitively, most semantic and shape information from the image can be encoded in the edges
 - More compact than pixels
- **Ideal:** artist's line drawing (but artist is also using object-level knowledge)



Why do we care about edges?

- Extract information, recognize objects

- Recover geometry and viewpoint



Origins of edges



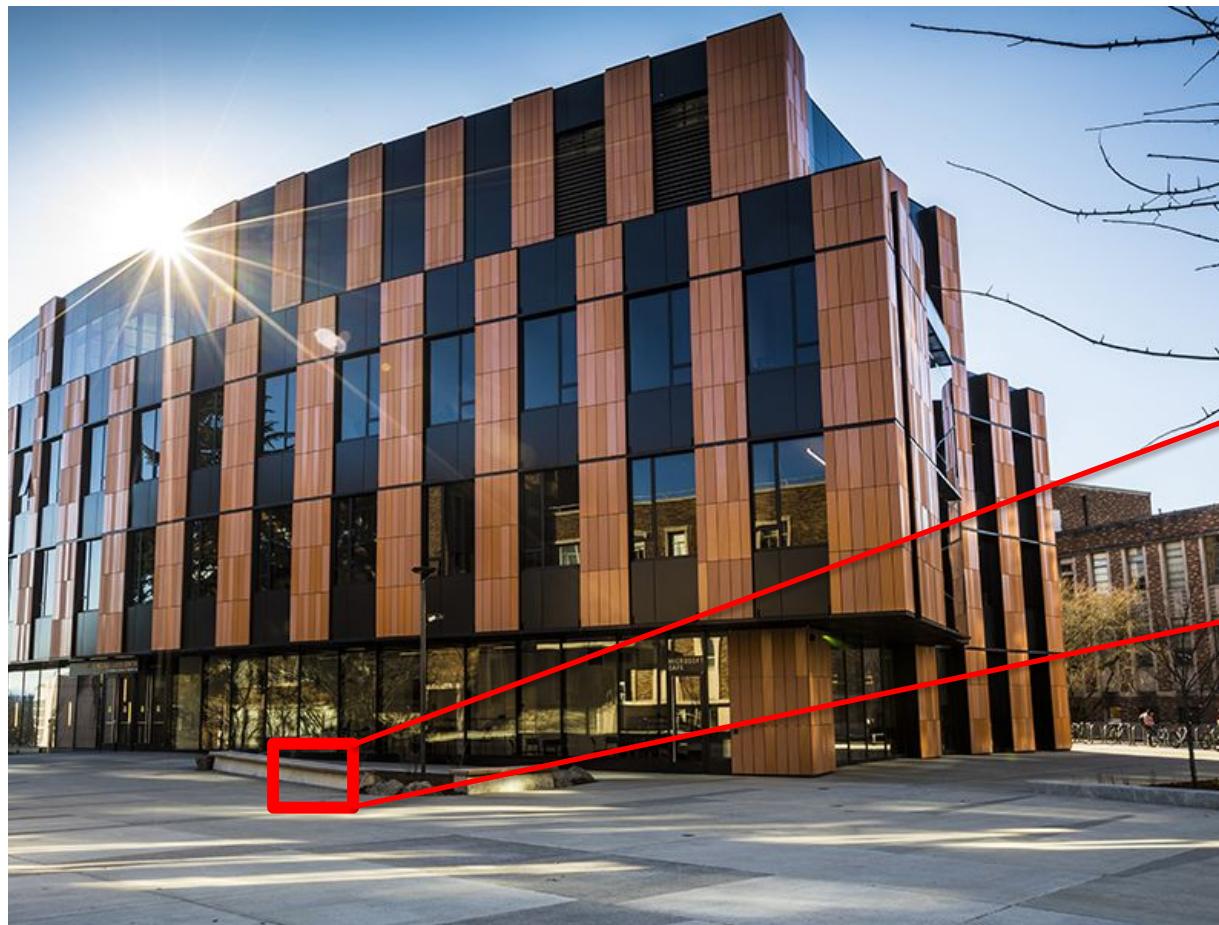
surface normal discontinuity

depth discontinuity

surface color discontinuity

illumination discontinuity

Closeup of edges



Surface normal discontinuity



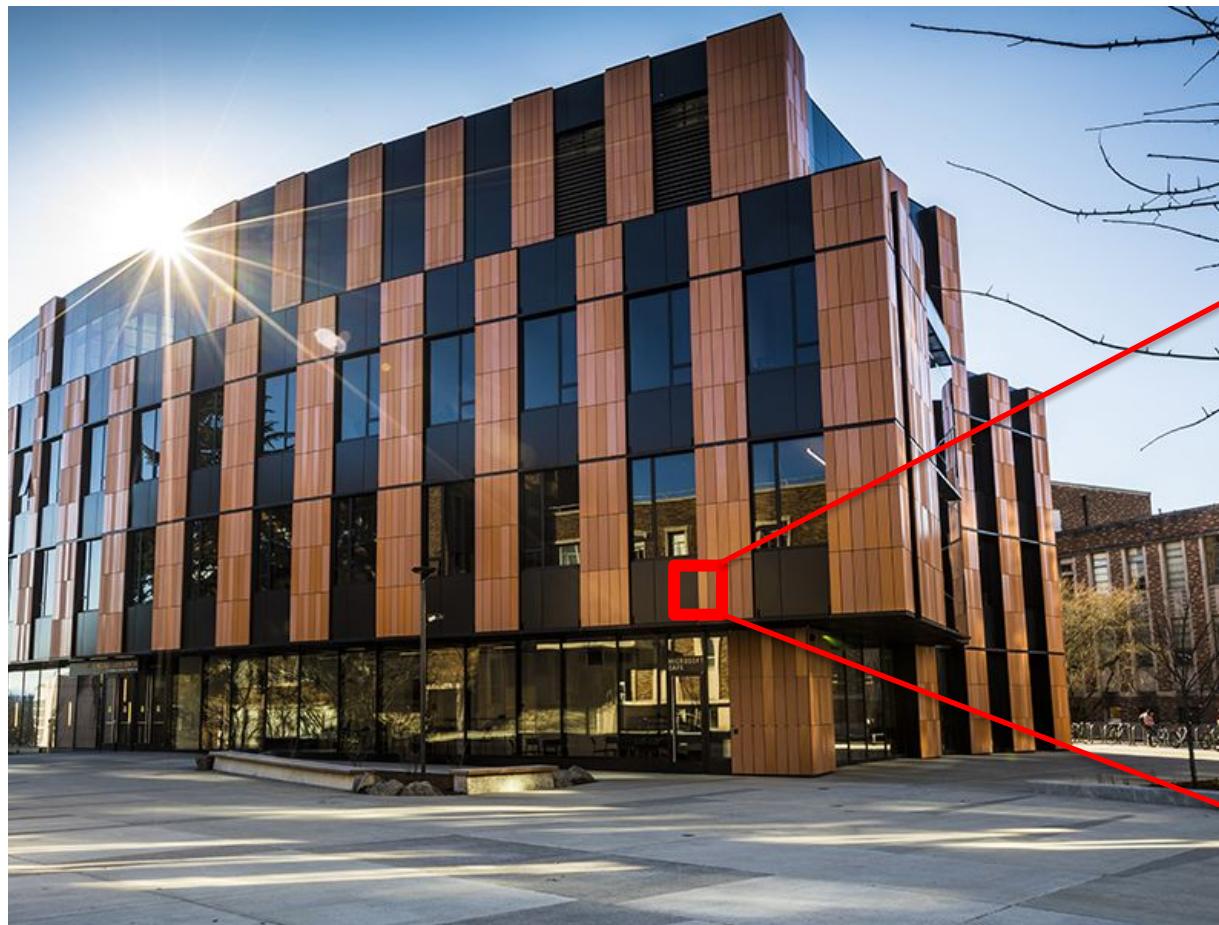
Closeup of edges



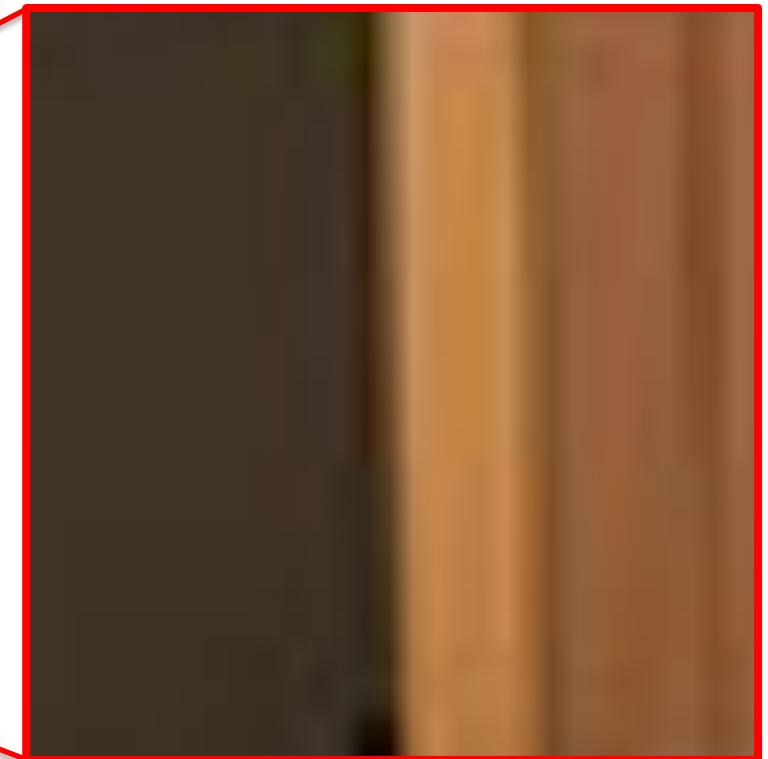
Depth discontinuity



Closeup of edges



Surface color discontinuity



What we will learn today

- Edge detection
- Image Gradients
- A simple edge detector
- Sobel 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

$$\begin{aligned}\frac{df}{dx} &= f[x] - f[x - 1] && \text{Backward} \\ &= f[x + 1] - f[x] && \text{Forward} \\ &= \frac{1}{2}(f[x + 1] - f[x - 1]) && \text{Central}\end{aligned}$$

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:

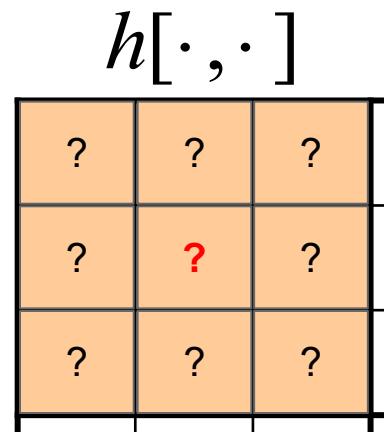
$$\frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

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:

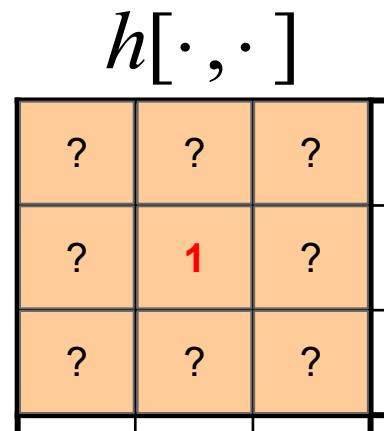
$$\frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

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

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

?	?	?
?	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	?	?
?	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:

$h[\cdot, \cdot]$

$\frac{1}{9}$	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:

$h[\cdot, \cdot]$

$\frac{1}{9}$	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:

$h[\cdot, \cdot]$

$\frac{1}{9}$	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
0	1	-1
0	0	0

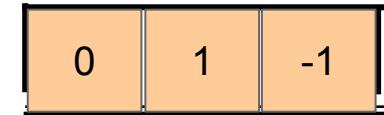
Remember the moving average filter:

$h[\cdot, \cdot]$

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

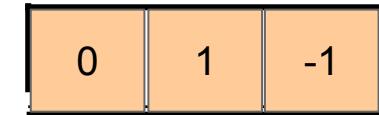
$$g[n, m] = f[n, m] - f[n, m - 1]$$

- Using Forward differentiation:

Q. What is the formula?

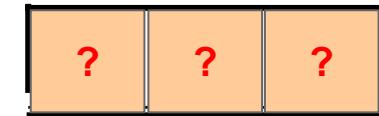
Designing filters that perform differentiation

- Using Backward differentiation:



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

- Using Forward differentiation:

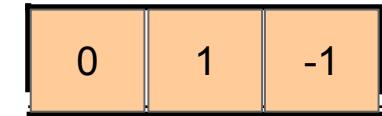


$$g[n, m] = f[n, m + 1] - f[n, m]$$

Q. What is the filter look like?

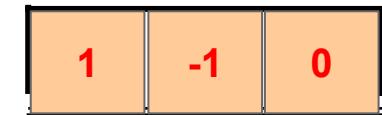
Designing filters that perform differentiation

- Using Backward differentiation:



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

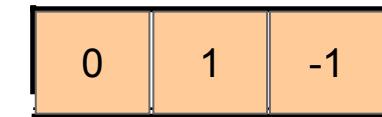
- Using Forward differentiation:



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

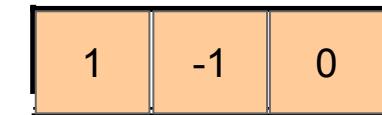
Designing filters that perform differentiation

- Using Backward differentiation:



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

- Using Forward differentiation:



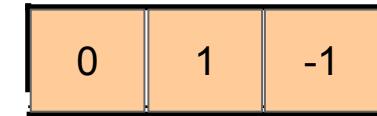
$$g[n, m] = f[n, m + 1] - f[n, m]$$

- Using Central differentiation:

Q. What is the formula?

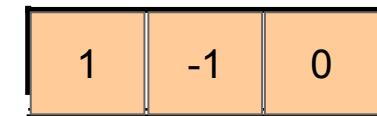
Designing filters that perform differentiation

- Using Backward differentiation:



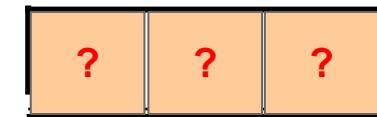
$$g[n, m] = f[n, m] - f[n, m - 1]$$

- Using Forward differentiation:



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

- Using Central differentiation:

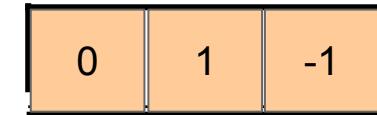


Q. What is
the filter?

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

Designing filters that perform differentiation

- Using Backward differentiation:



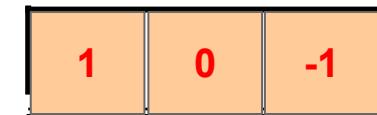
$$g[n, m] = f[n, m] - f[n, m - 1]$$

- 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
---	---	----

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

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

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

Derivative in width dimension for one row

Using backward differentiation:

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

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

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

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

Derivative in width dimension for one row

Using backward differentiation:

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

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

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

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

Derivative in width dimension for one row

Using backward differentiation:

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

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

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

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

Derivative in width dimension for one row

Using backward differentiation:

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

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

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

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

Derivative in width dimension for one row

Using backward differentiation:

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

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

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

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

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 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?

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

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

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

2D discrete derivative - example

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

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

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

2D discrete derivative - example

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

Let's do the other one

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

Let's do the other one

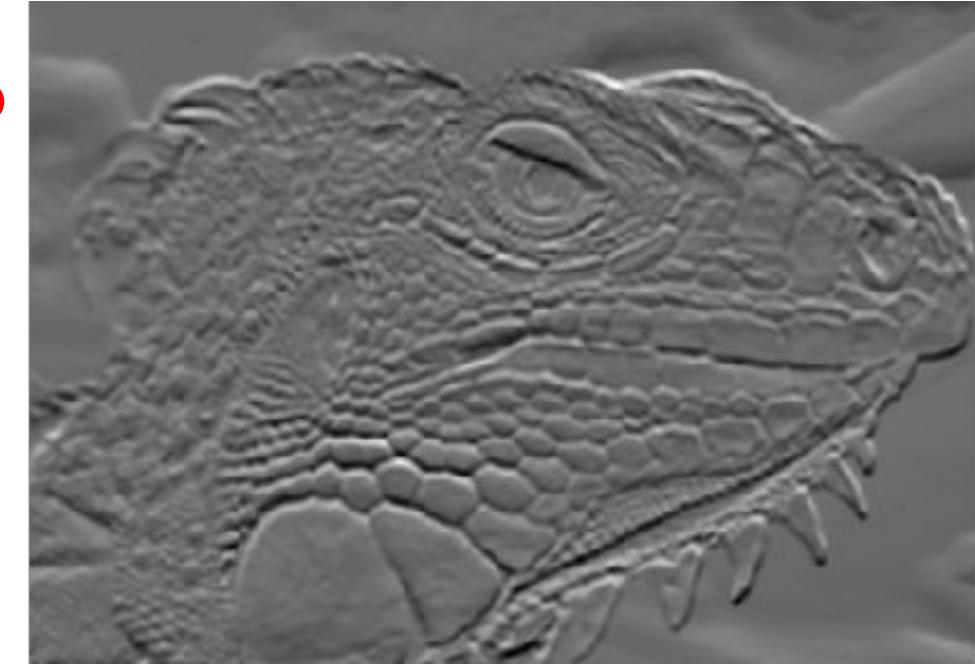
$$f[n, m] = \begin{bmatrix} 10 & 10 & 20 & 20 & 20 \\ 10 & 10 & 20 & 20 & 20 \\ 10 & 10 & 20 & 20 & 20 \\ 10 & 10 & 20 & 20 & 20 \\ 10 & 10 & 20 & 20 & 20 \end{bmatrix} \quad h[n, m] = [1 \quad 0 \quad -1]$$
$$g[n, m] = \begin{bmatrix} 10 & 10 & 10 & 0 & -20 \\ 10 & 10 & 10 & 0 & -20 \\ 10 & 10 & 10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

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



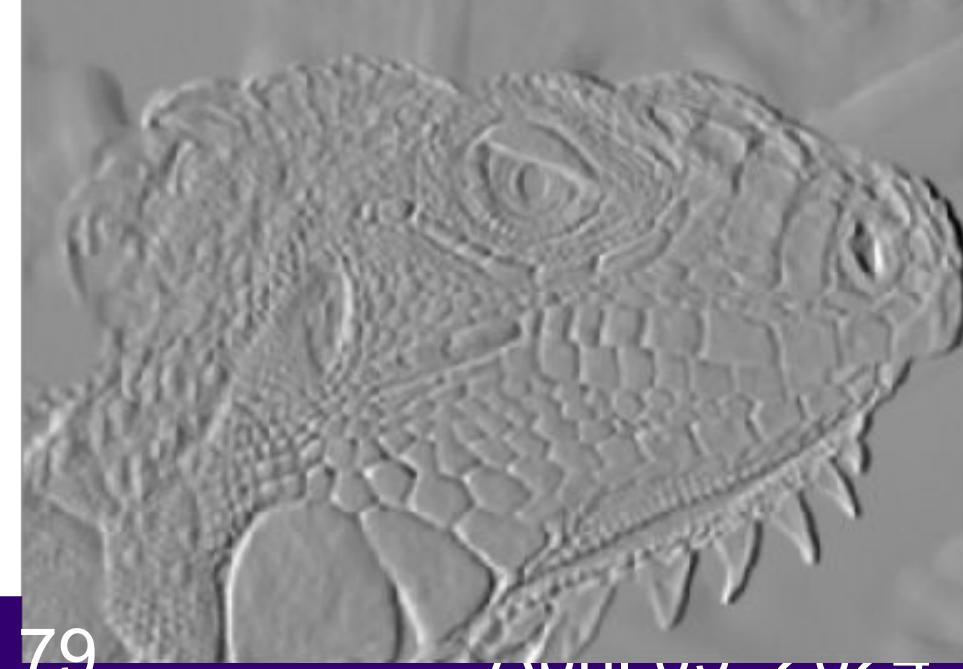
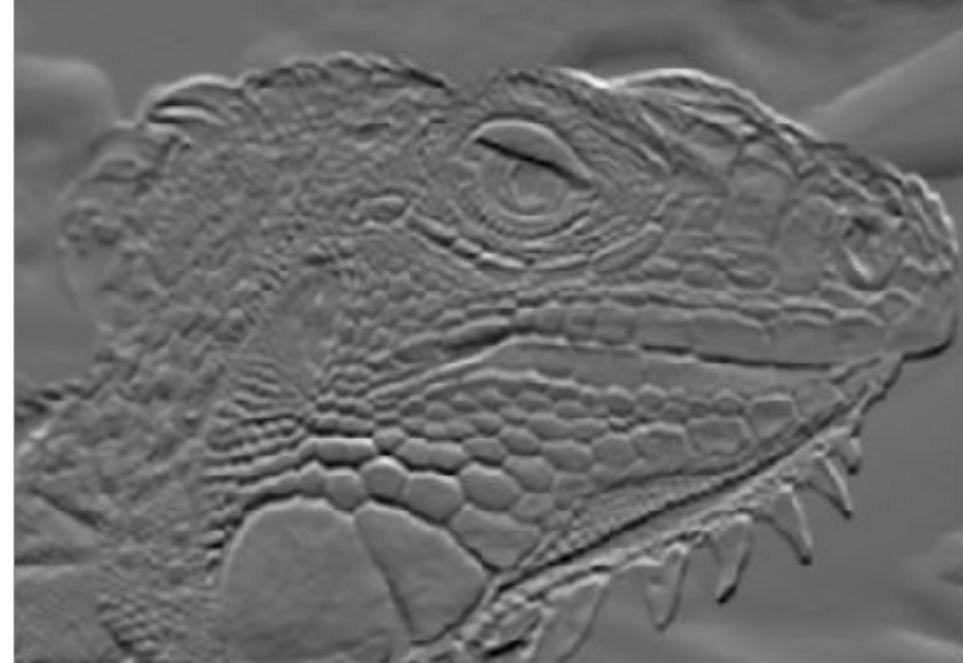
Q. Which filter was applied?

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

A

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

B



What we will learn today

- Edge detection
- Image Gradients
- A simple edge detector
- Sobel edge detector

Characterizing edges

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

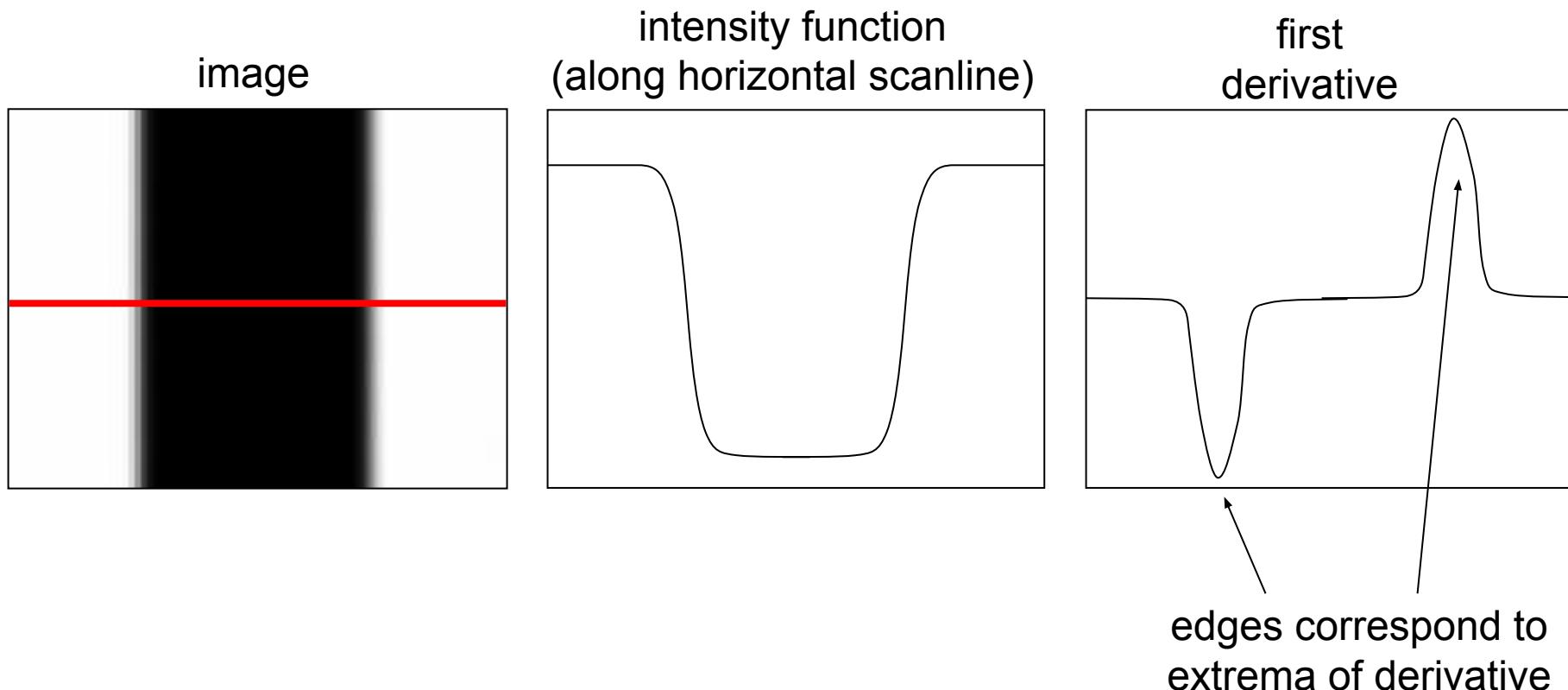
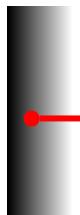
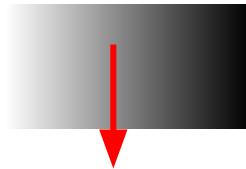


Image gradient

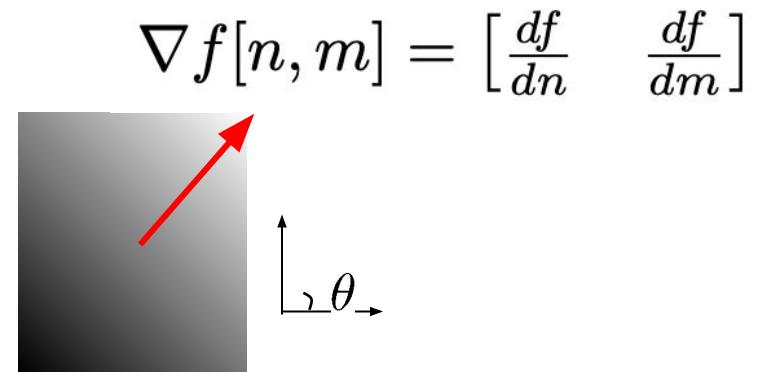
The gradient of an image:



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



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

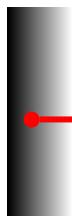


The gradient vector points in the direction of most rapid increase in intensity

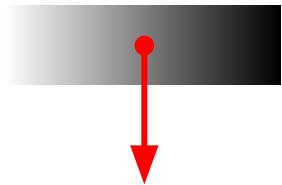
$$\theta = \tan^{-1}\left(\frac{f_m}{f_n}\right)$$

Image gradient

The gradient of an image:

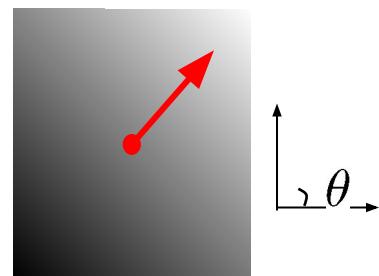


$$\nabla_m f[n, m] = \begin{bmatrix} 0 & \frac{df}{dm} \end{bmatrix}$$



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

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



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

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

The *edge strength* is given by the gradient magnitude

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

Finite differences: example

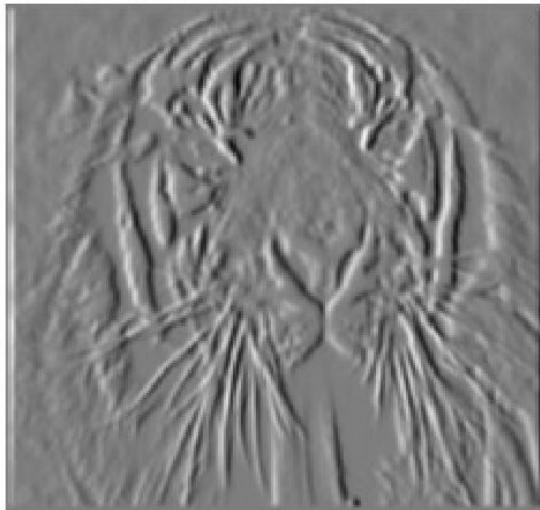
Original
Image



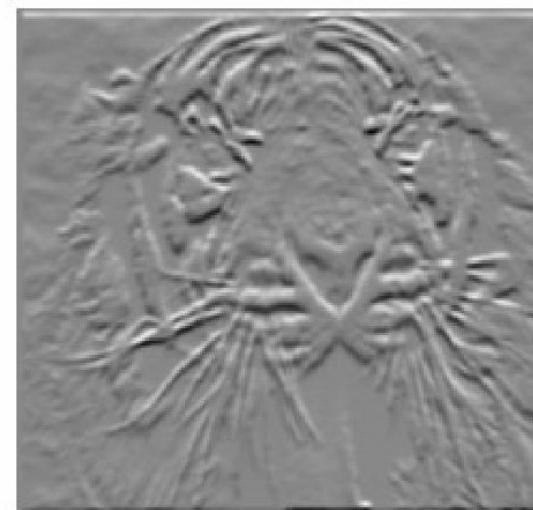
Gradient
magnitude



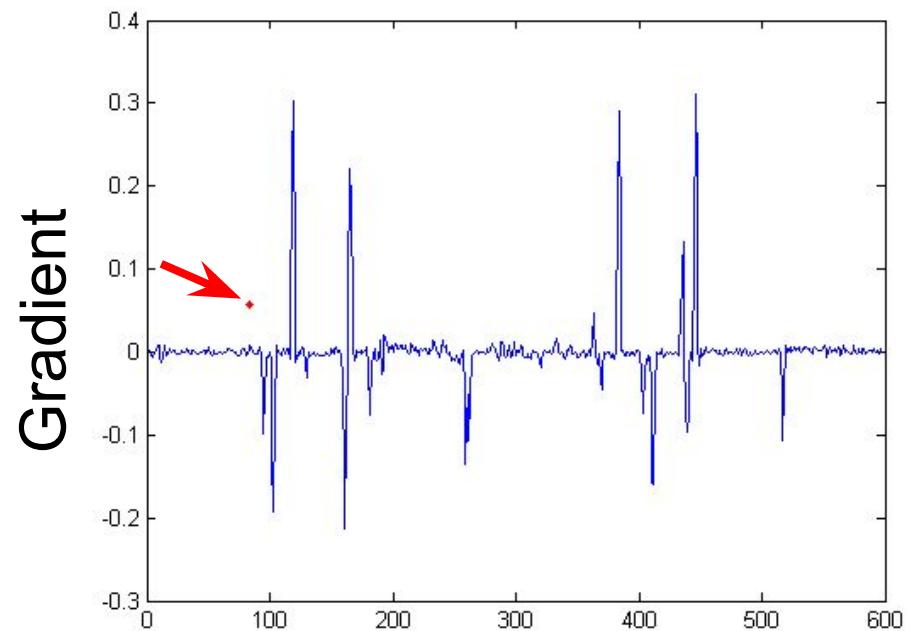
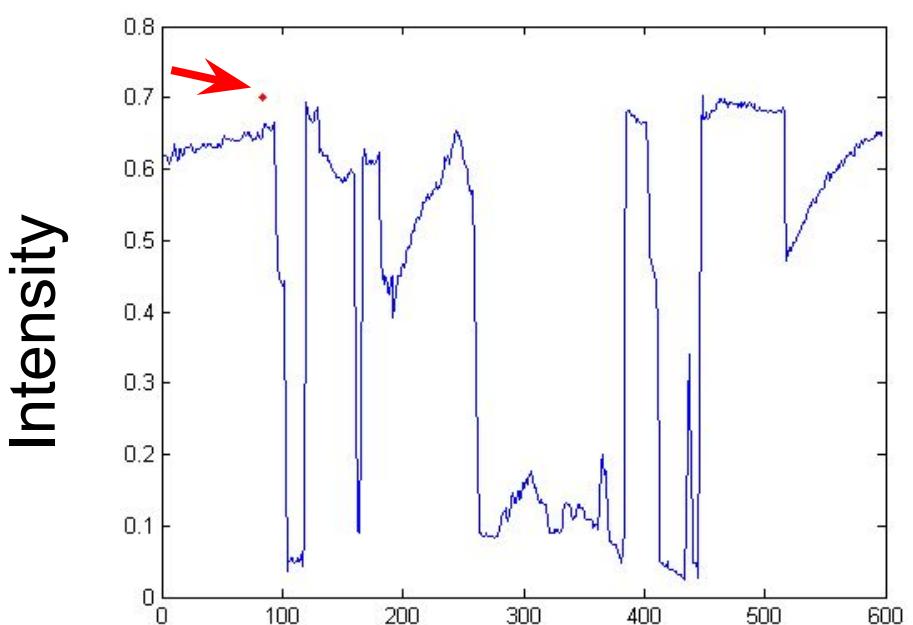
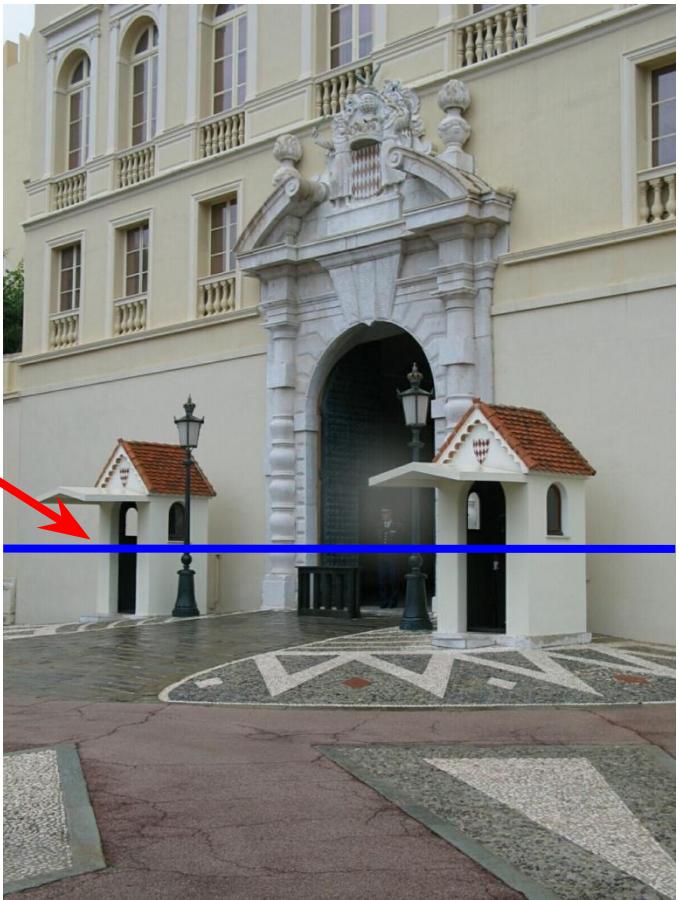
width-direction



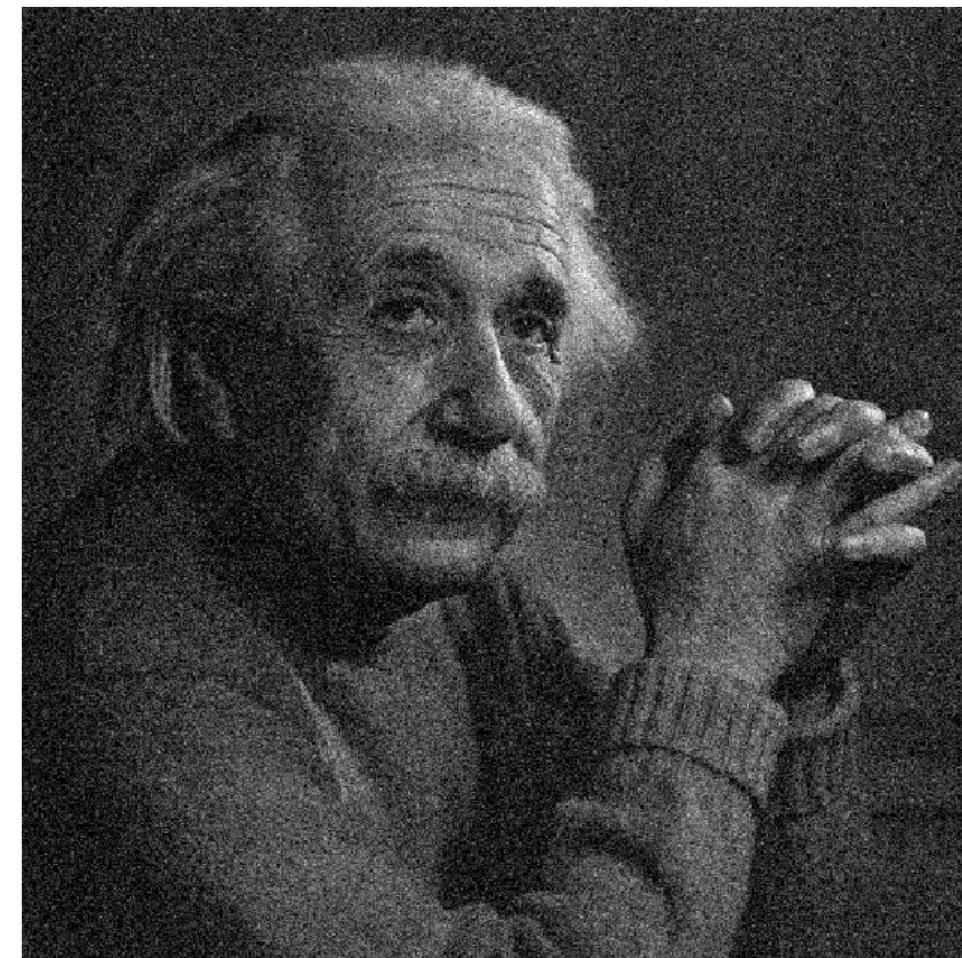
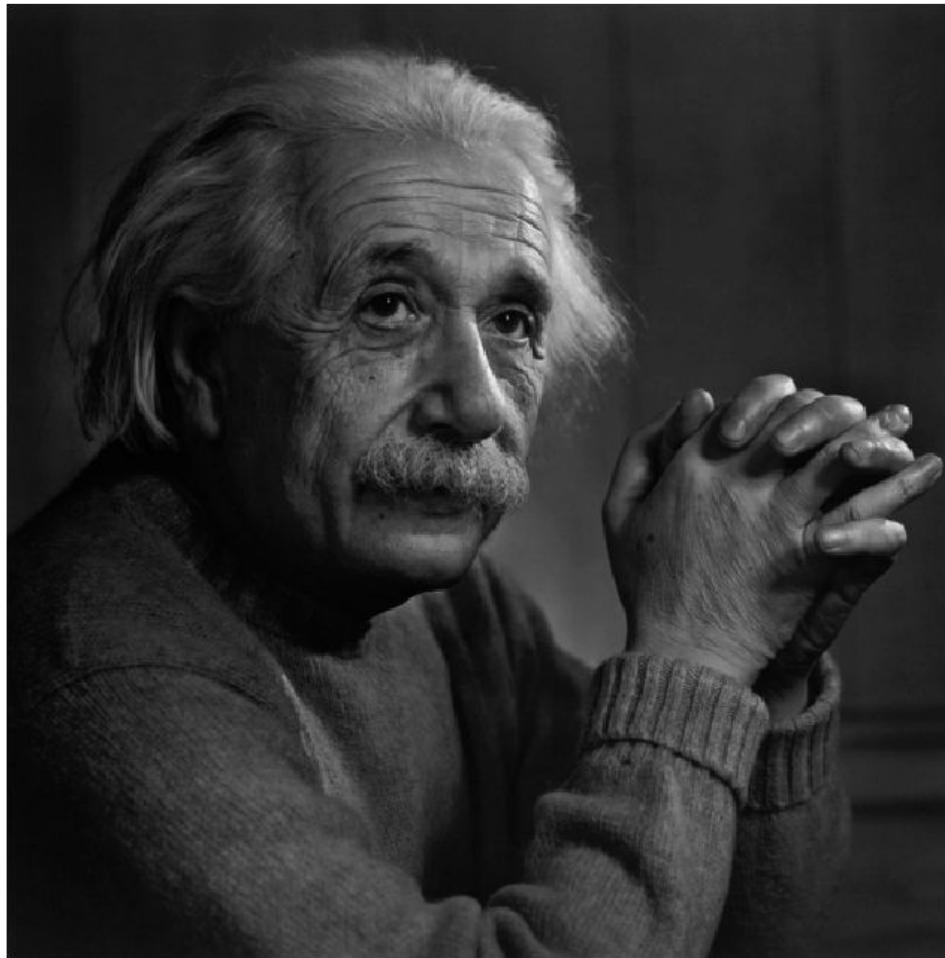
height-direction



Intensity profile

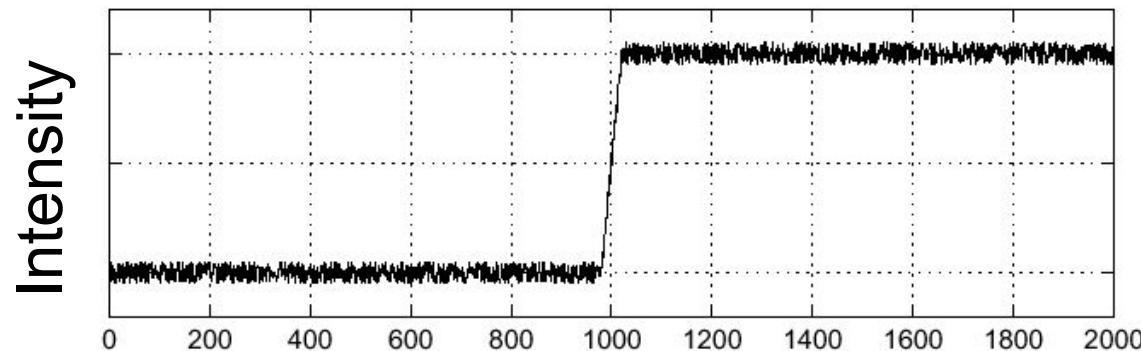


Q. What will happen if we use this edge detector on a noisy pixels?



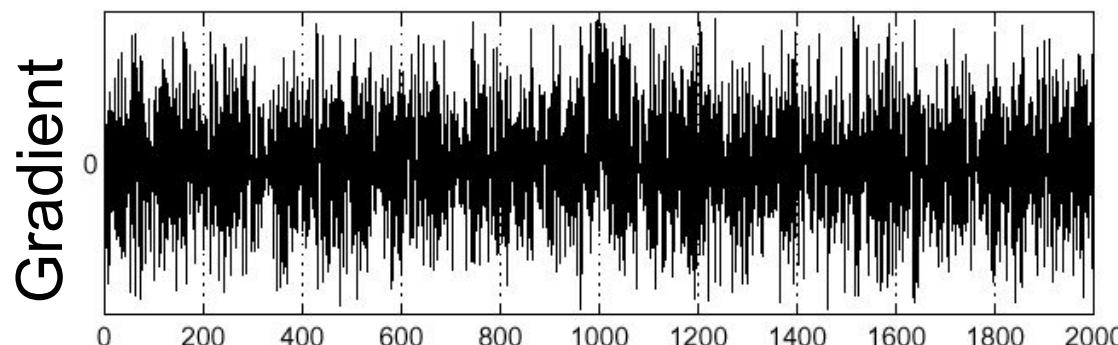
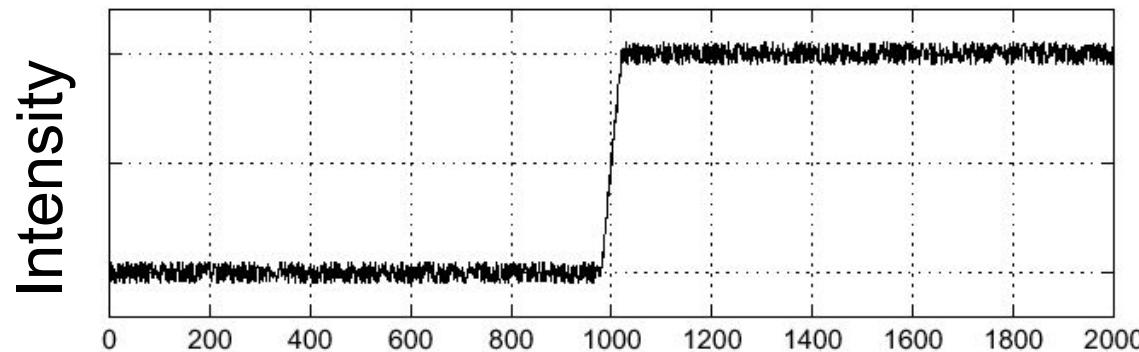
Effects of noise

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



Effects of noise

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



Where is the edge?

Effects of noise

- Finite difference filters respond strongly to noise
 - Image noise results in pixels that look very different from their neighbors
 - Generally, the larger the noise the stronger the response
- Q. What is a potential quick fix for noisy images?

Effects of noise

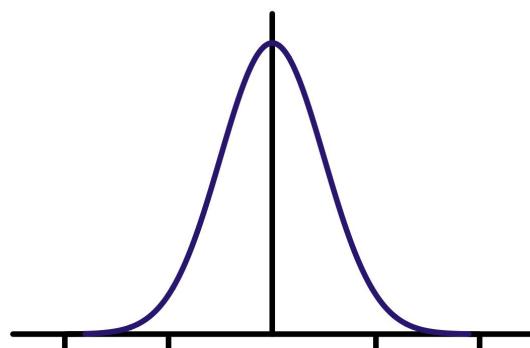
- Finite difference filters respond strongly to noise
 - Image noise results in pixels that look very different from their neighbors
 - Generally, the larger the noise the stronger the response
- Q. What is a potential quick fix for noisy images?
- Smoothing the image should help, by forcing pixels different to their neighbors (=noise pixels?) to look more like neighbors

Smoothing with different filters

- Mean smoothing

$$\frac{1}{3} \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \quad \frac{1}{3} [1 \ 1 \ 1]$$

- Gaussian (smoothing * derivative)



$$\frac{1}{4} \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix}$$

$$\frac{1}{4} [1 \ 2 \ 1]$$

Smoothing with different filters

3x3



Mean



Gaussian



Median

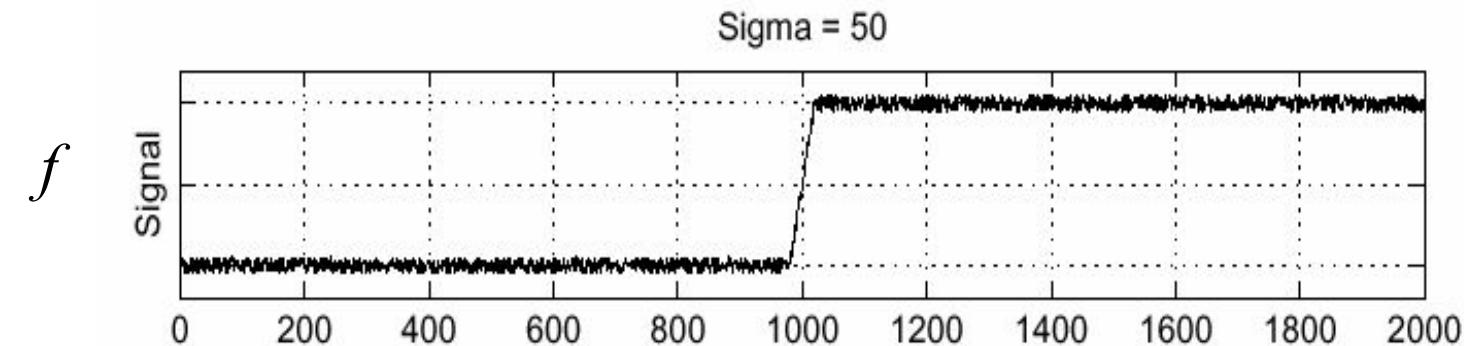
5x5



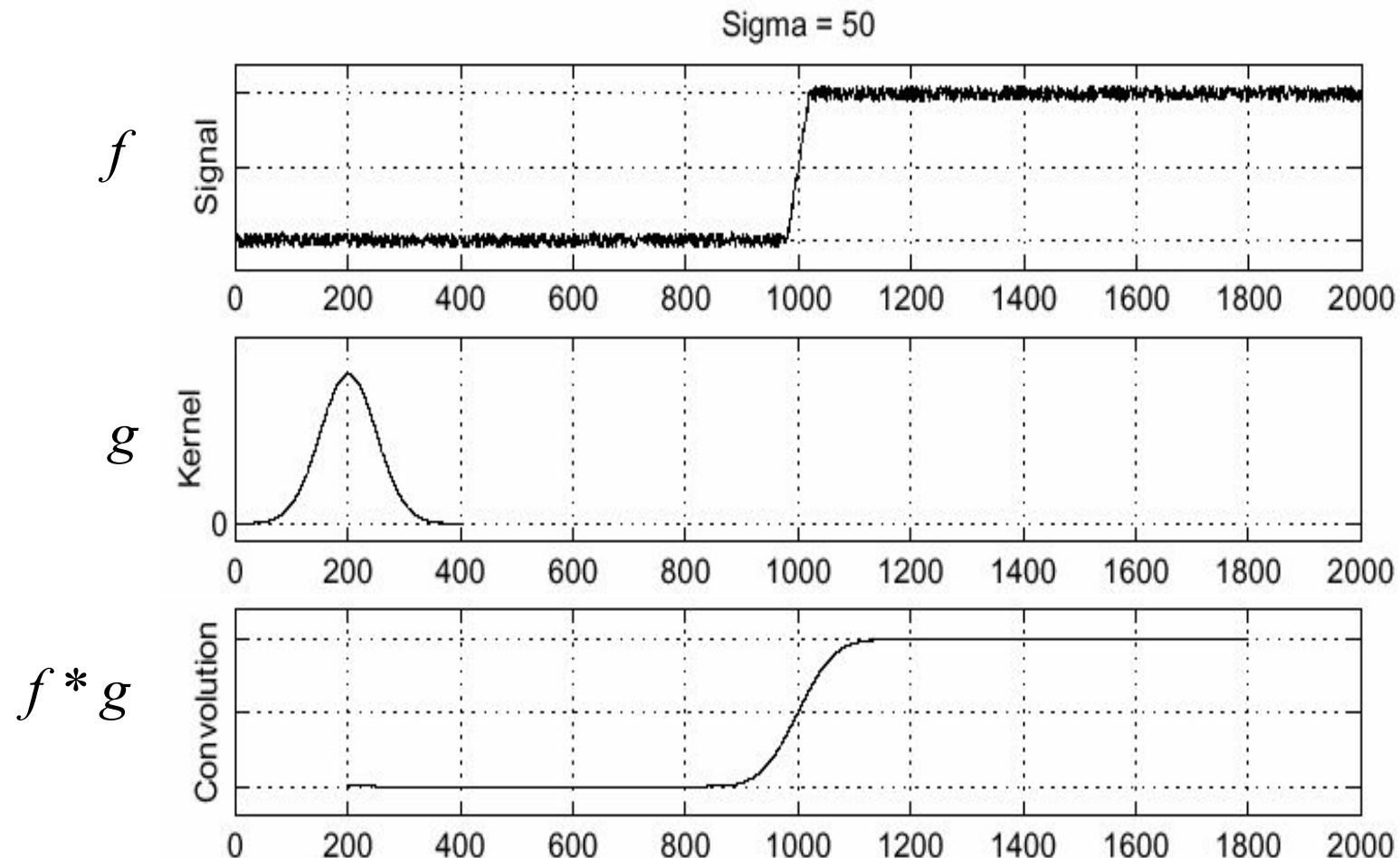
7x7



Solution: input function



Solution: smooth first



Solution: smooth first

To find edges, look for peaks in

$$\frac{d}{dx}(f * g)$$

$$\frac{d}{dx}(f * g)$$

f

g

$f * g$

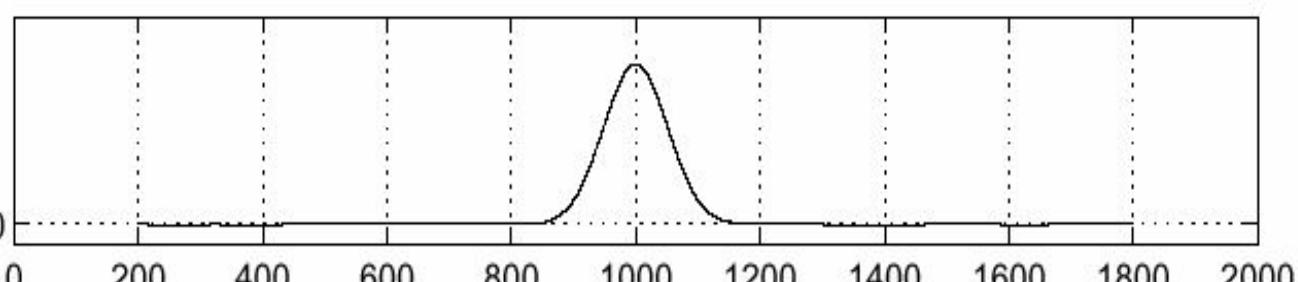
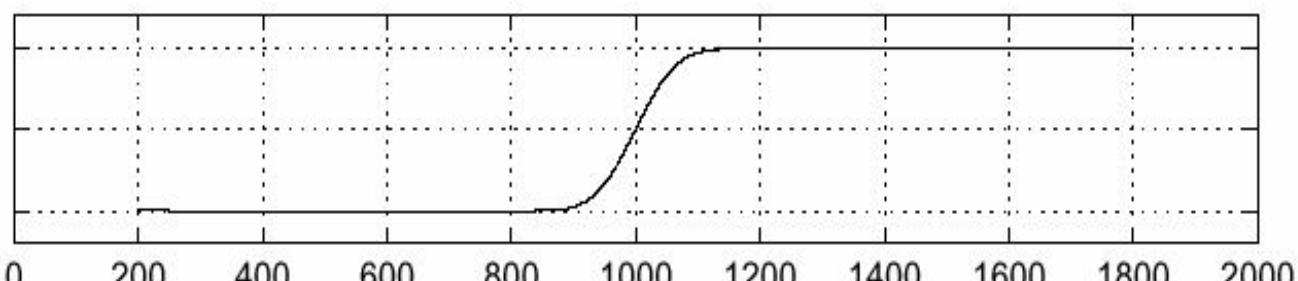
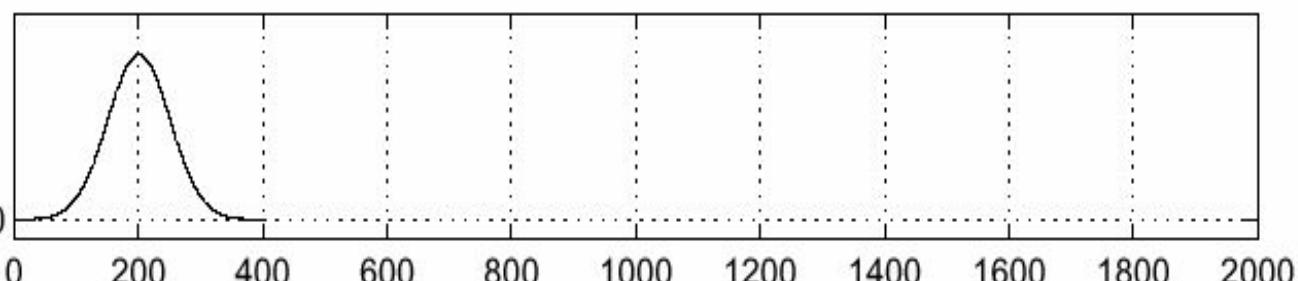
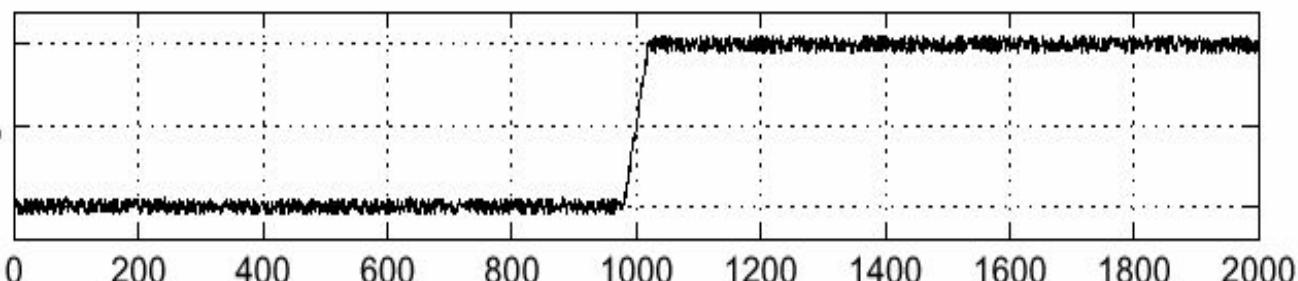
Differentiation

Signal

Kernel

Convolution

Sigma = 50



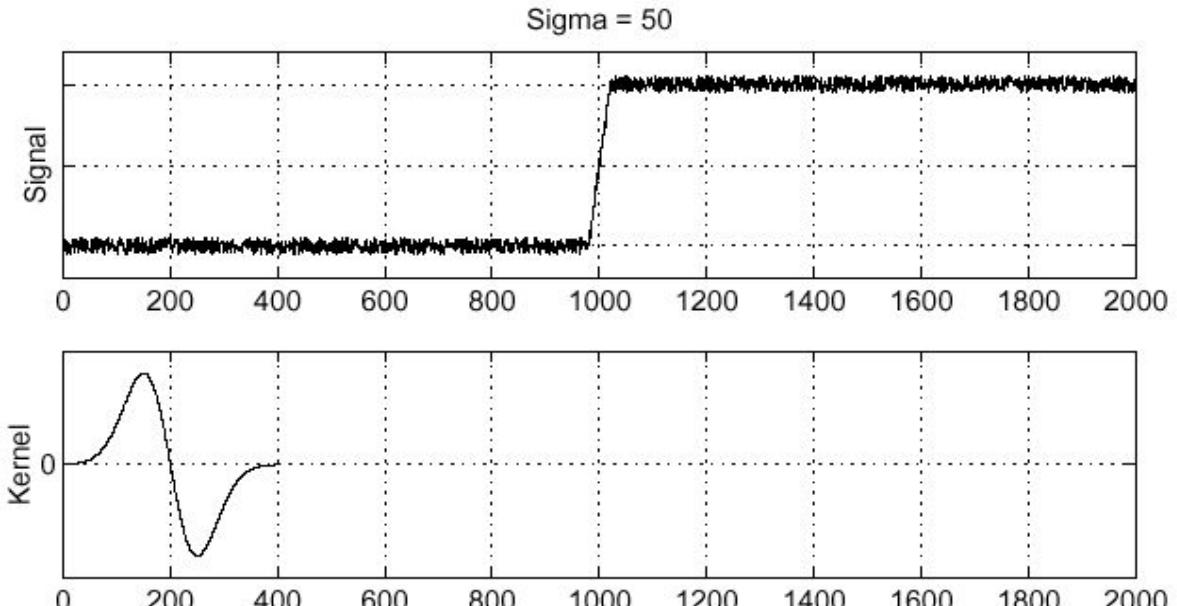
Derivative theorem of convolution

- This theorem gives us a very useful property:

$$\frac{d}{dx}(f * g) = f * \frac{d}{dx}g$$

f

$$\frac{d}{dx}g$$



Derivative theorem of convolution

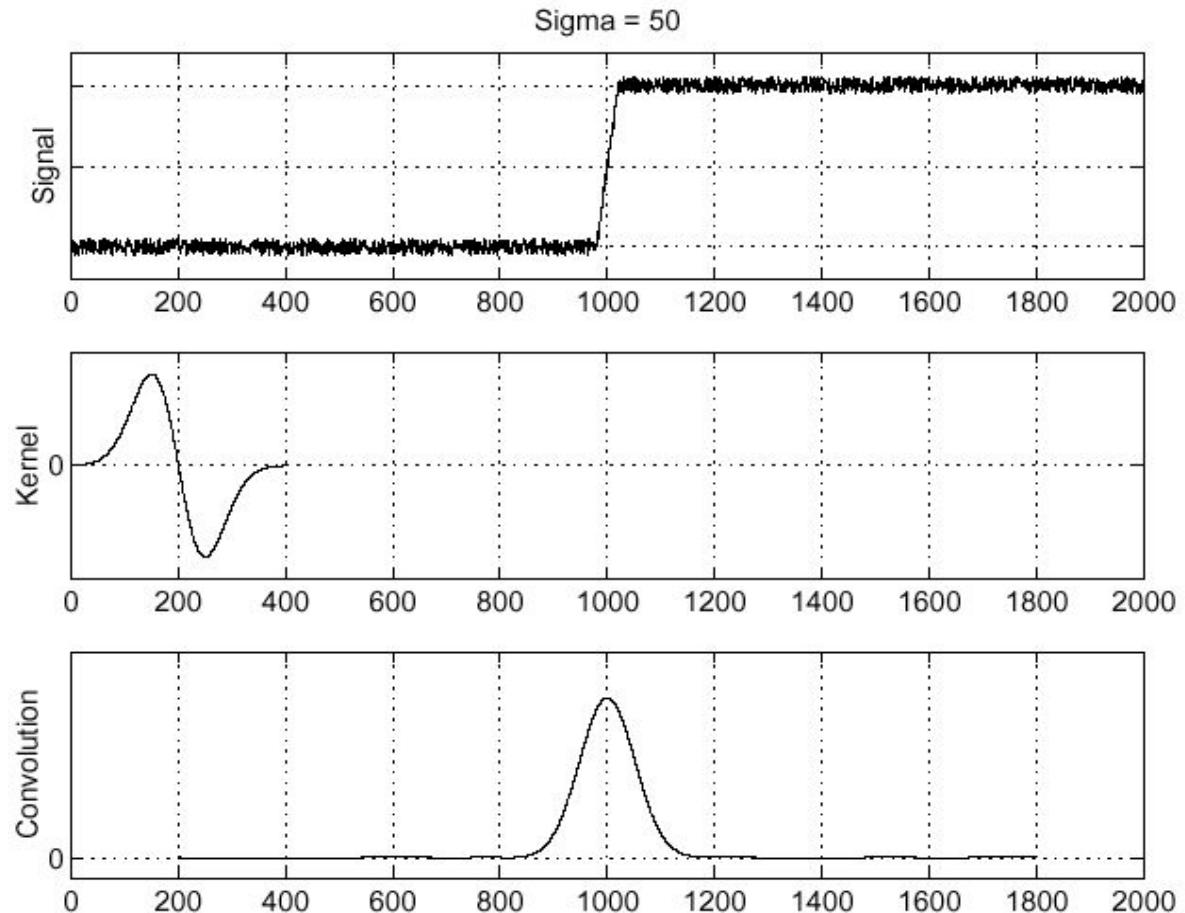
- This theorem gives us a very useful property:

$$\frac{d}{dx}(f * g) = f * \frac{d}{dx}g$$

- This saves us one operation:

We can precompute:

$$f * \frac{d}{dx}g$$



Derivative theorem of convolution

- This theorem gives us a very useful property:

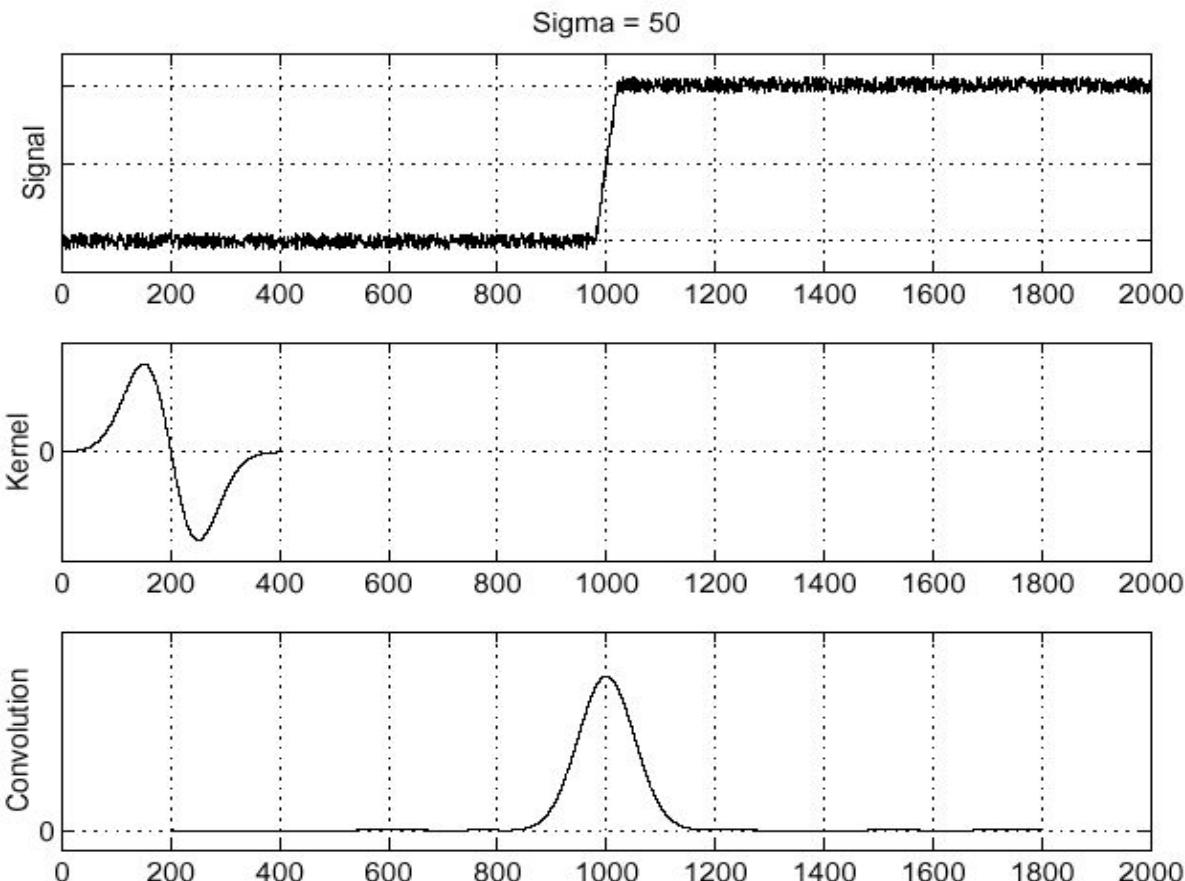
$$\frac{d}{dx}(f * g) = f * \frac{d}{dx}g$$

- This saves us one operation:

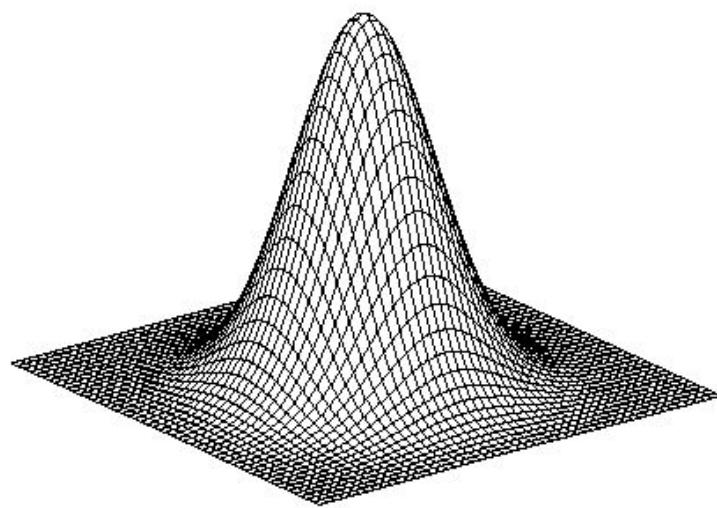
f

$$\frac{d}{dx}g$$

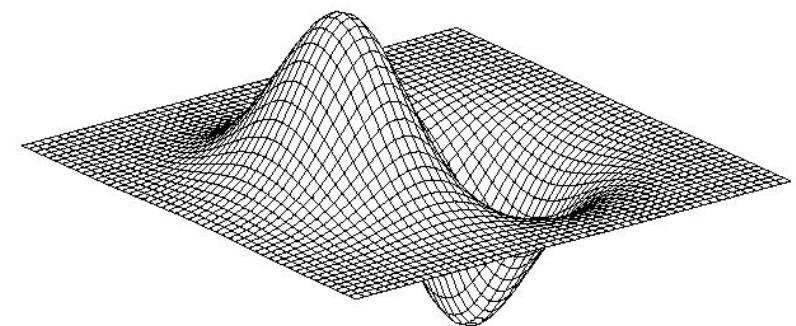
$$f * \frac{d}{dx}g$$



Derivative of Gaussian filter (central derivative)



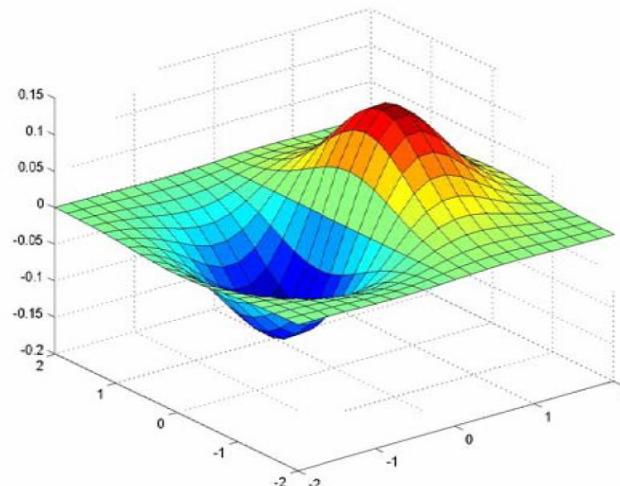
$$* \begin{bmatrix} 1 & 0 & -1 \end{bmatrix} =$$



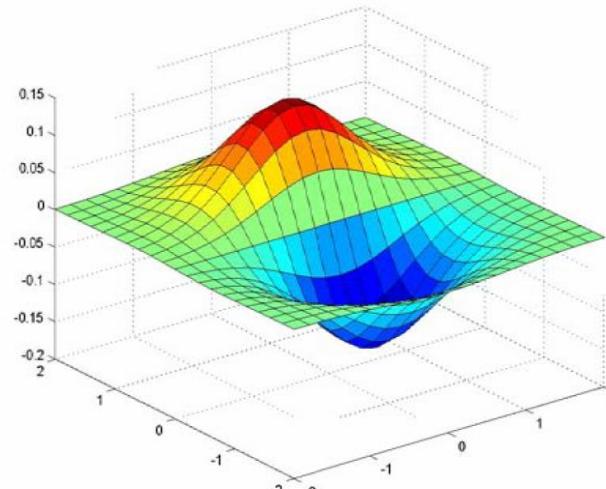
2D-gaussian

x - derivative

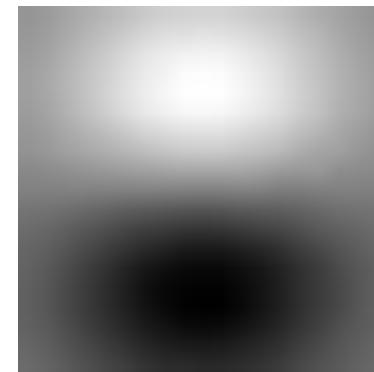
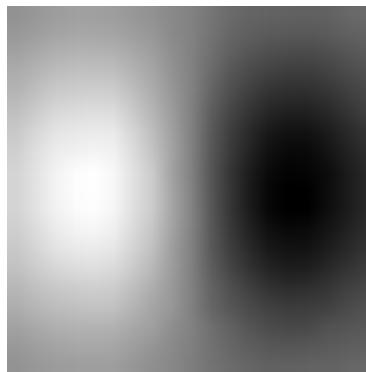
Derivative of Gaussian filter



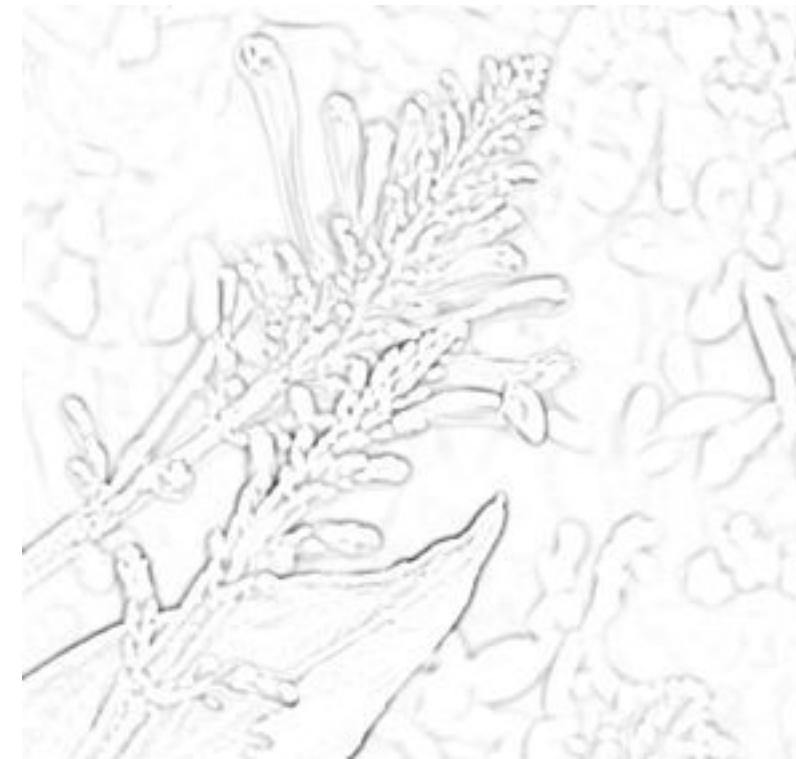
x-direction



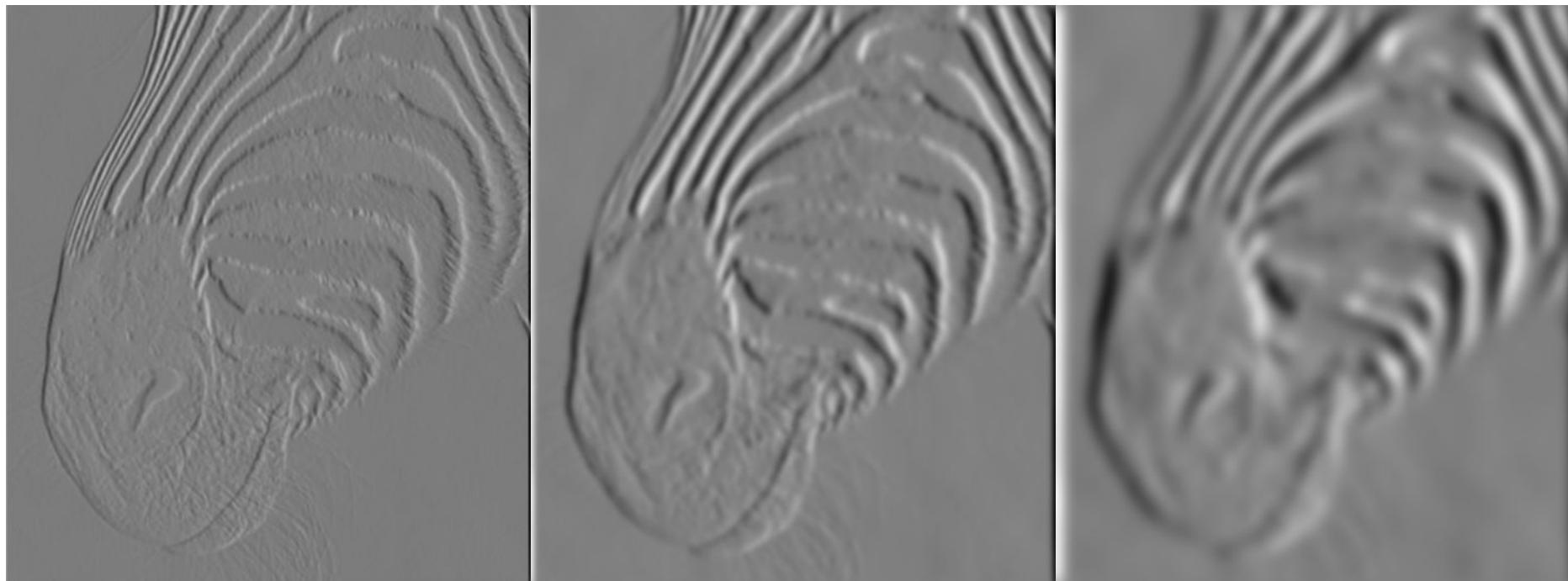
y-direction



Derivative of Gaussian filter



Tradeoff between smoothing at different scales



1 pixel

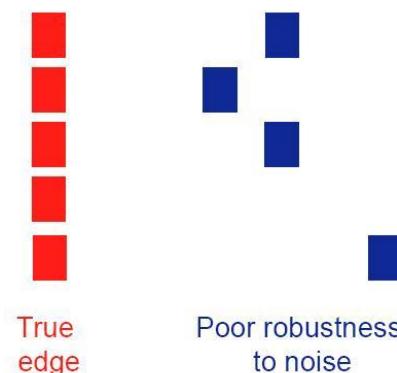
3 pixels

7 pixels

Smoothed derivative removes noise, but blurs edge.
Also finds edges at different “scales”.

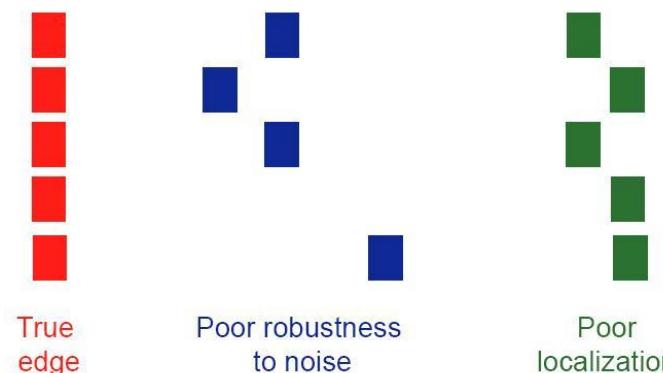
Designing an edge detector

- Criteria for an “optimal” edge detector:
 - **Good detection:** the optimal detector must minimize the probability of false positives (detecting spurious edges caused by noise), as well as that of false negatives (missing real edges)



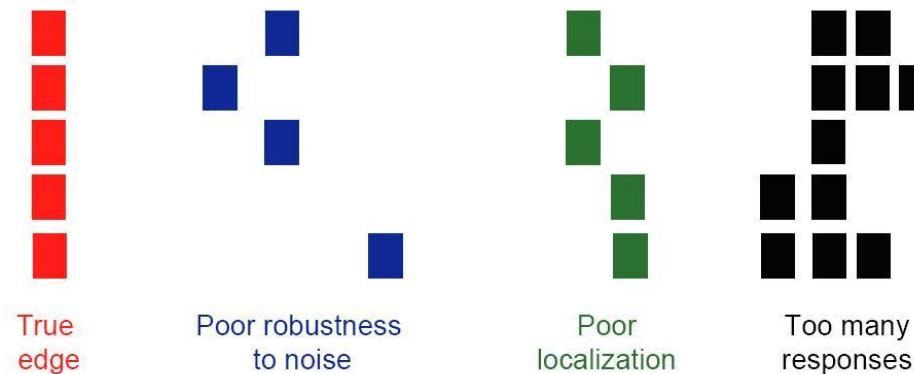
Designing an edge detector

- Criteria for an “optimal” edge detector:
 - **Good detection:** the optimal detector must minimize the probability of false positives (detecting spurious edges caused by noise), as well as that of false negatives (missing real edges)
 - **Good localization:** the edges detected must be as close as possible to the true edges



Designing an edge detector

- Criteria for an “optimal” edge detector:
 - **Good detection:** the optimal detector must minimize the probability of false positives (detecting spurious edges caused by noise), as well as that of false negatives (missing real edges)
 - **Good localization:** the edges detected must be as close as possible to the true edges
 - **Single response:** the detector must return one point only for each true edge point; that is, minimize the number of local maxima around the true edge



What we will learn today

- Edge detection
- Image Gradients
- A simple edge detector
- Sobel Edge detector

Sobel Operator

- uses two 3×3 kernels which are convolved with the original image to calculate approximations of the derivatives
- one for horizontal changes, and one for vertical

$$\mathbf{G}_x = \begin{bmatrix} +1 & 0 & -1 \\ +2 & 0 & -2 \\ +1 & 0 & -1 \end{bmatrix} \quad \mathbf{G}_y = \begin{bmatrix} +1 & +2 & +1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

Sobel Operation

- Smoothing + differentiation

$$\mathbf{G}_x = \begin{bmatrix} +1 & 0 & -1 \\ +2 & 0 & -2 \\ +1 & 0 & -1 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} [+1 \quad 0 \quad -1]$$

Gaussian smoothing differentiation

The diagram illustrates the decomposition of the Sobel operator \mathbf{G}_x into Gaussian smoothing and differentiation components. On the left, the operator is represented as a 3x3 matrix. To its right is a column vector representing Gaussian smoothing. An arrow points from the middle row of the matrix to the vector, labeled "Gaussian smoothing". To the right of the vector is a scalar value (2) and a second column vector representing differentiation. Another arrow points from the middle row of the matrix to this second vector, labeled "differentiation".

Sobel Operation

- Magnitude:

$$\mathbf{G} = \sqrt{\mathbf{G}_x^2 + \mathbf{G}_y^2}$$

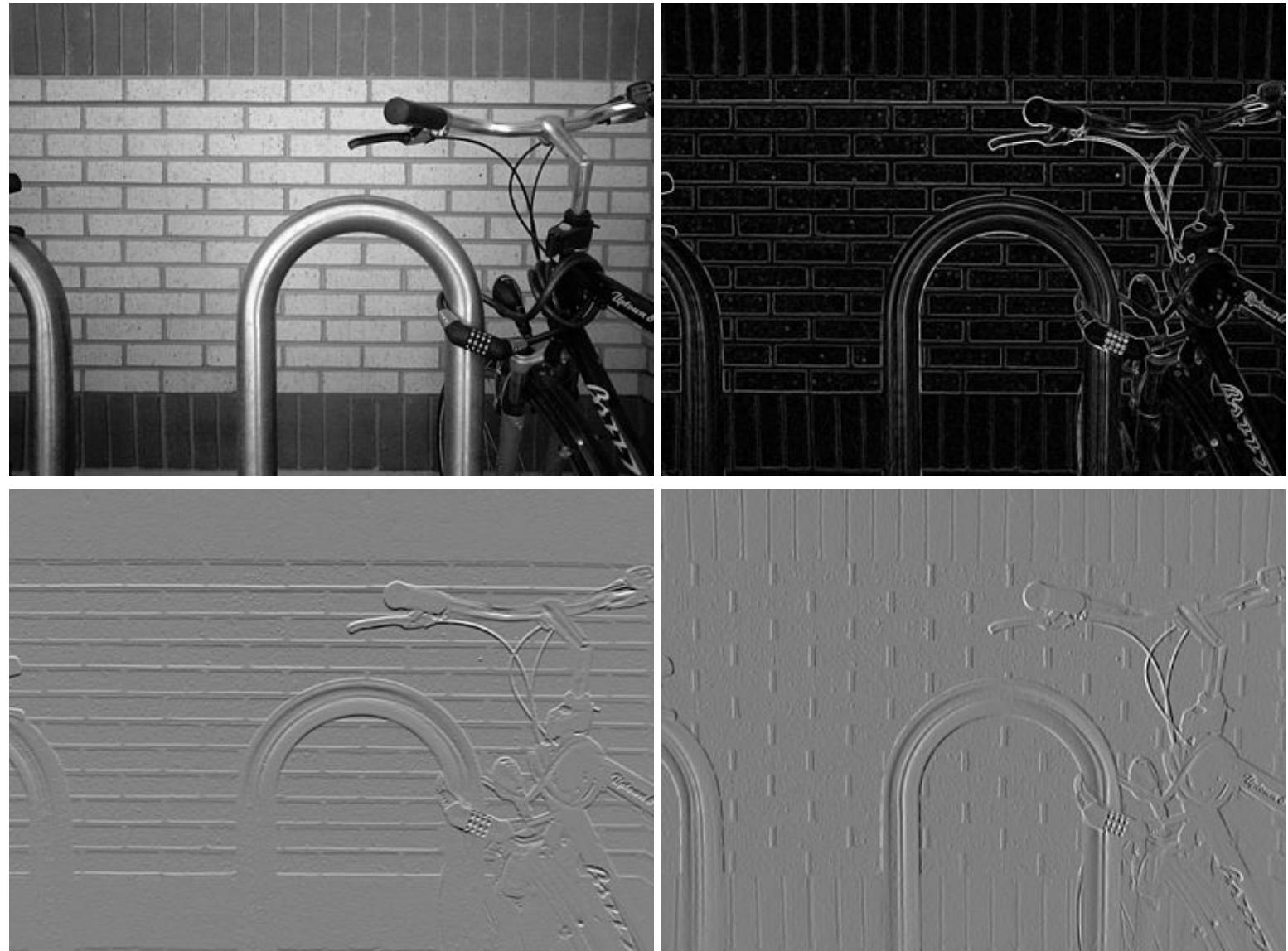
- Angle or direction of the gradient:

$$\Theta = \text{atan}\left(\frac{\mathbf{G}_y}{\mathbf{G}_x}\right)$$

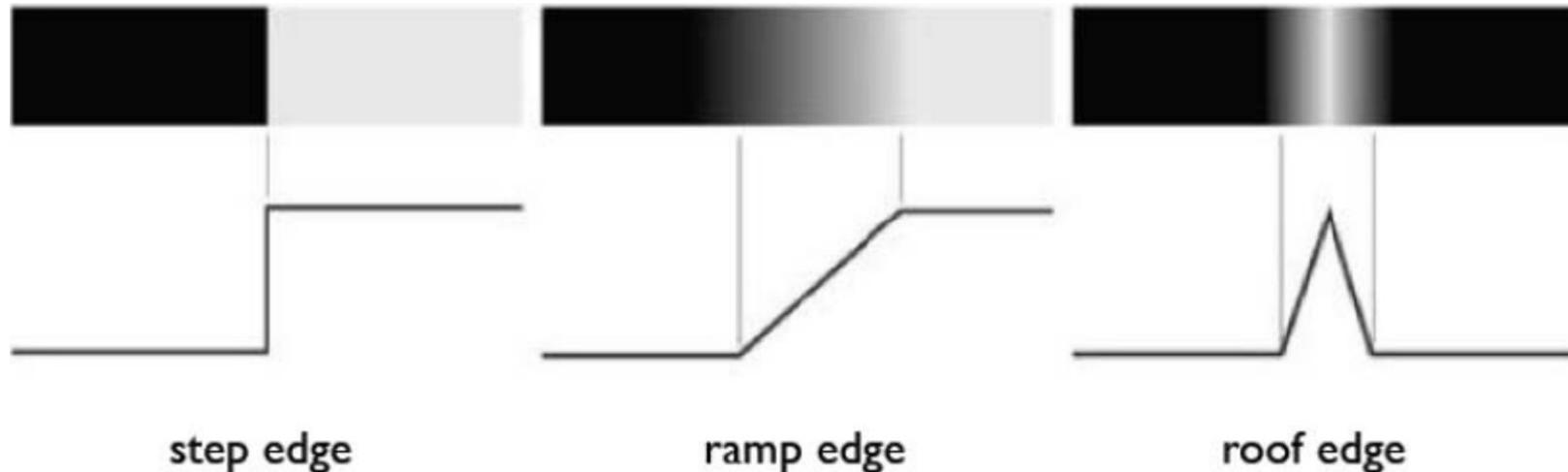
Sobel Filter example

Step 1: Calculate the gradient magnitude at every pixel location.

Step 2: Threshold the values to generate a binary image



Sobel Filter Problems



- Poor Localization (Trigger response in multiple adjacent pixels)
- Thresholding value favors certain directions over others
 - Can miss oblique edges more than horizontal or vertical edges
 - False negatives

What we will learn today

- Edge detection
- Image Gradients
- A simple edge detector
- Sobel Edge detector

Next time: Lines and features