

COMPUTER VISION LECTURE 7-8 – EDGE DETECTION

Prof. Dr. Francesco Maurelli
2019-09-24
2019-09-25



WHAT WE WILL LEARN TODAY

1. Edge detection
2. Image Gradients
3. A simple edge detector
4. Sobel edge detector
5. Canny edge detector



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

FILTER EXAMPLE #1: MOVING AVERAGE

This filter “Replaces” each pixel with an average of its neighborhood.

Achieve smoothing effect (remove sharp features)

$$h[\times, \times]$$

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

1	1	1
1	1	1
1	1	1

Original image



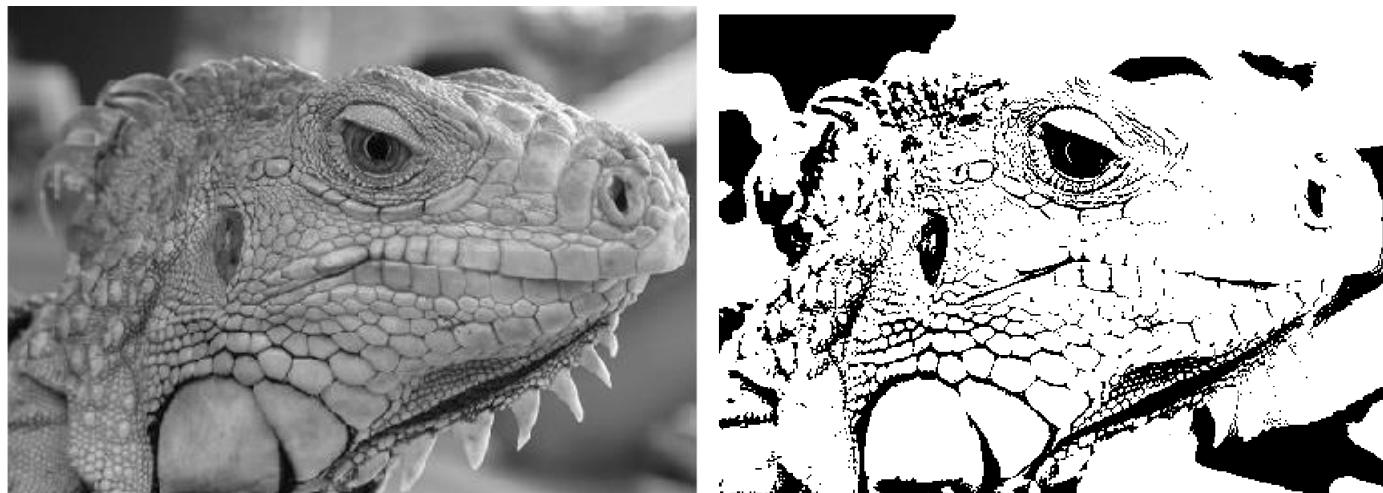
Smoothed image



FILTER EXAMPLE #2: IMAGE SEGMENTATION

Image segmentation based on a simple threshold:

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



FILTER EXAMPLE #3: SHARPENING

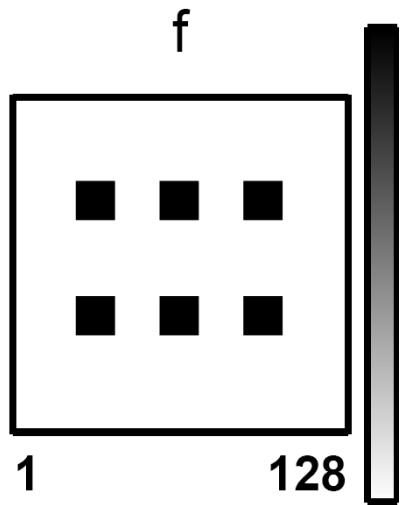


Cross correlation of two 2D signals $f[n,m]$ and $h[n,m]$:

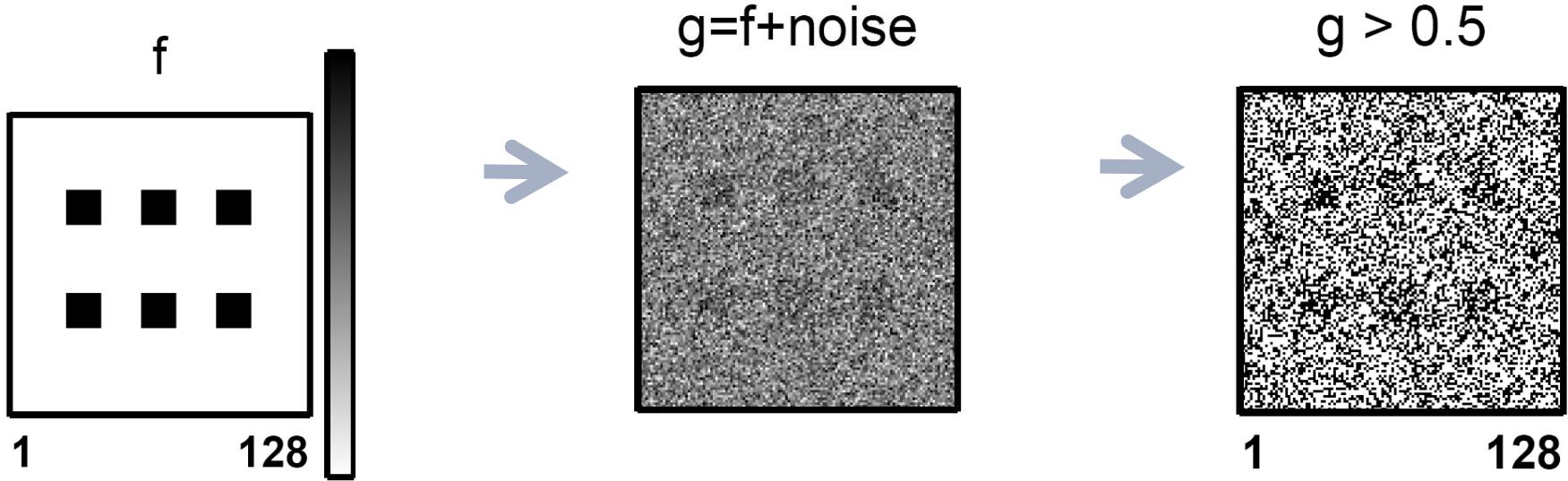
$$f[n, m] \text{ ** } h[n, m] = \sum_{k=-\infty}^{\infty} \sum_{l=-\infty}^{\infty} f[n, m] \times h[n + k, m + l]$$

Equivalent to a convolution without the flip

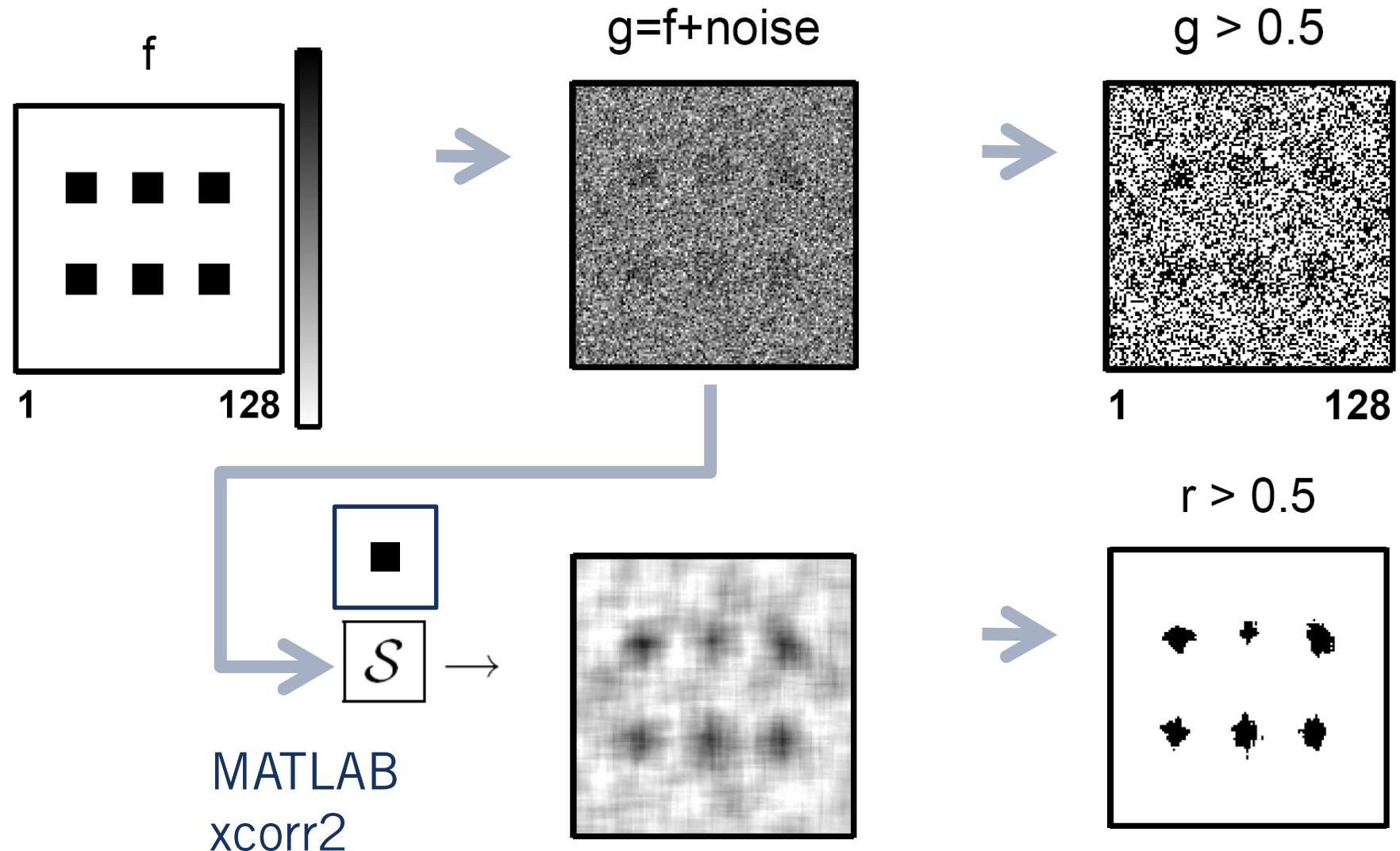
(CROSS) CORRELATION – EXAMPLE



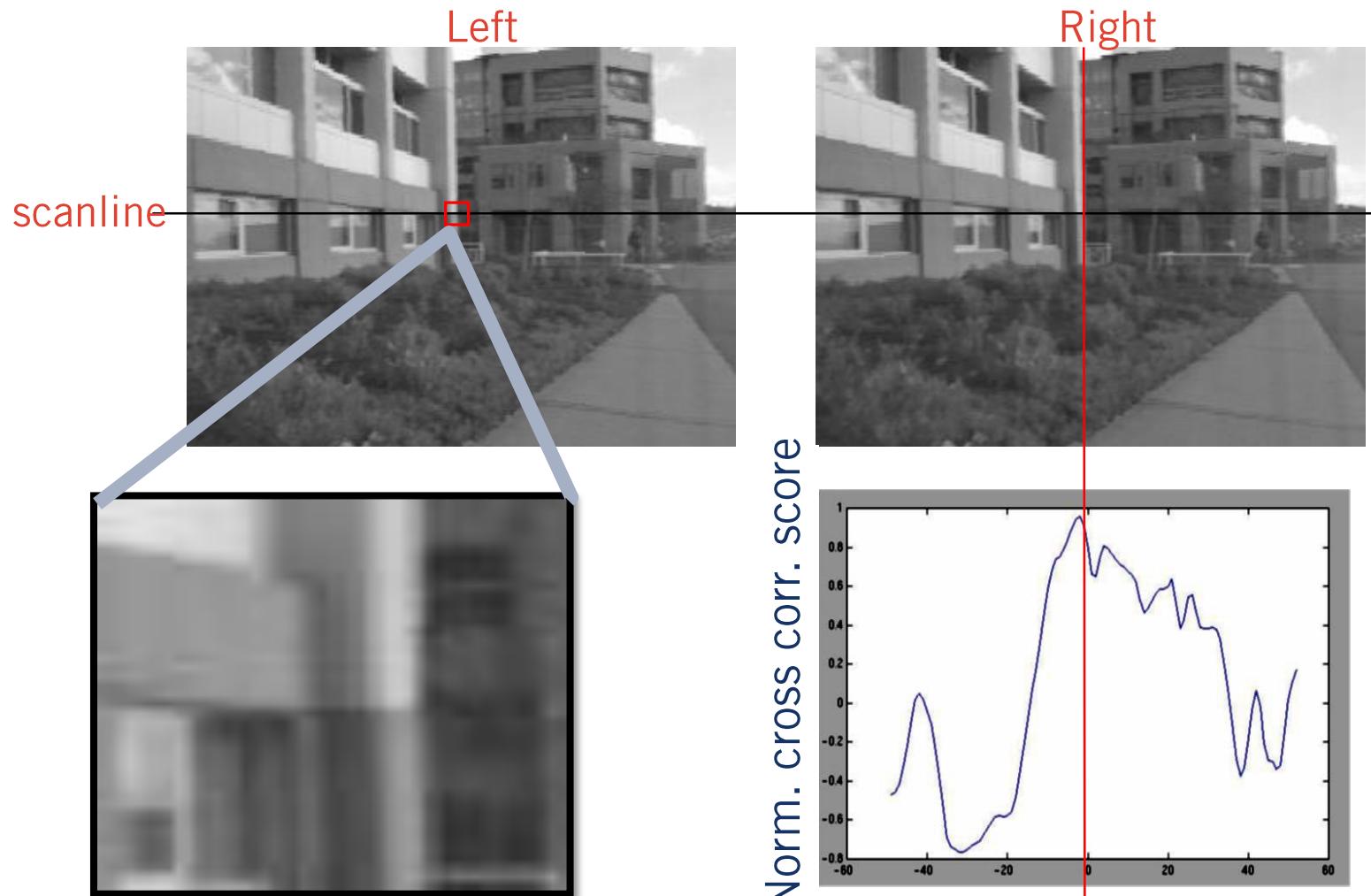
(CROSS) CORRELATION – EXAMPLE



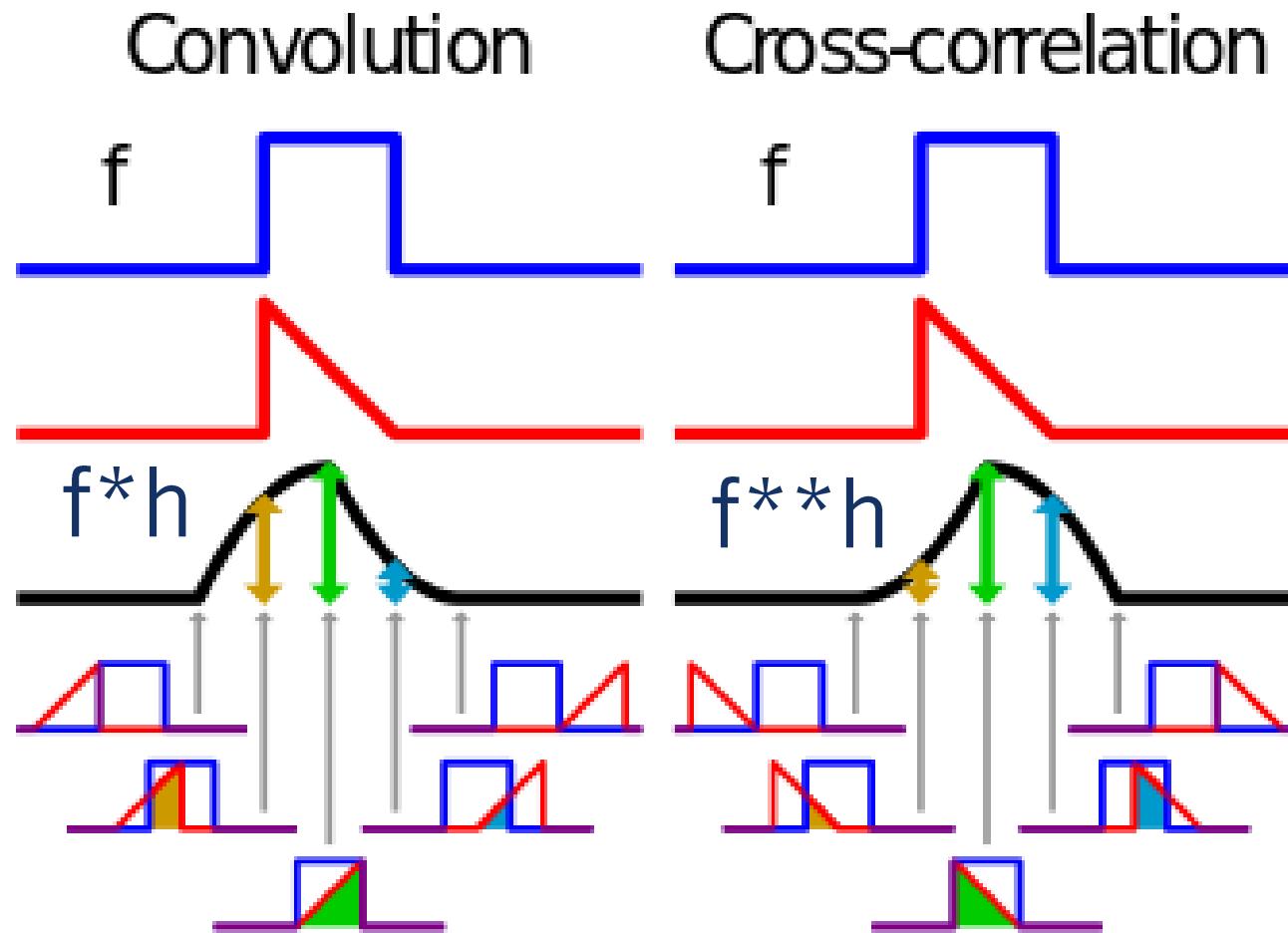
(CROSS) CORRELATION – EXAMPLE



(CROSS) CORRELATION – EXAMPLE



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



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.

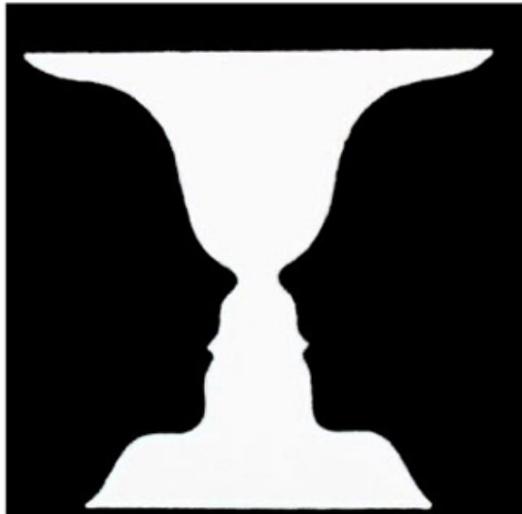
→ correlation is a measure of **relatedness** of two signals

WHAT WE WILL LEARN TODAY

1. Edge detection
2. Image Gradients
3. A simple edge detector
4. Sobel edge detector
5. Canny edge detector

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

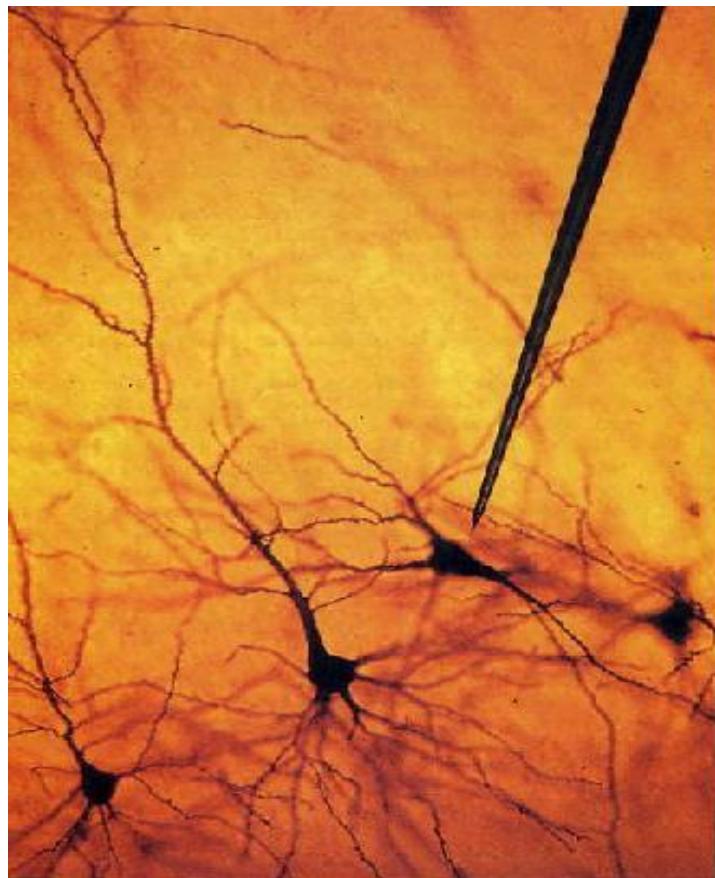
EDGE → MEANING?



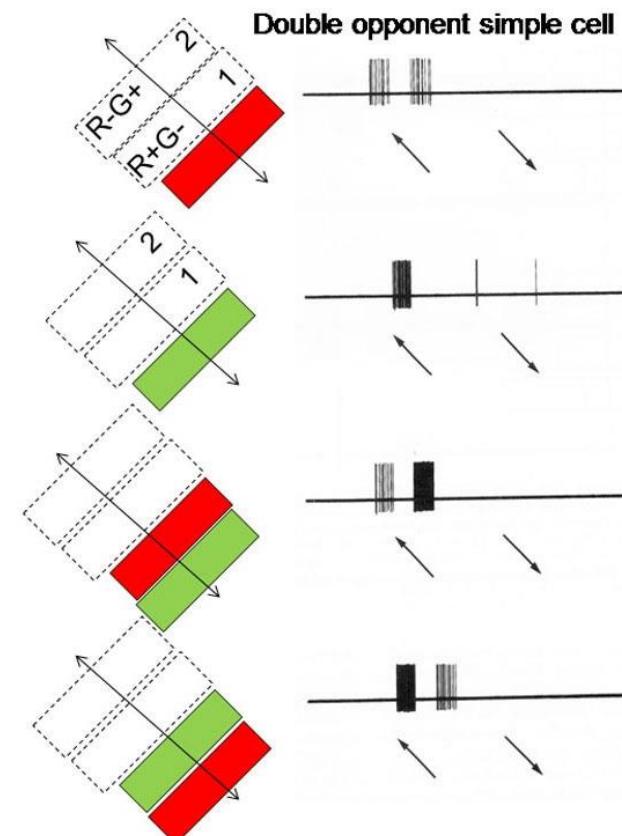


- (A) Cave painting at Chauvet, France, about 30,000 B.C.;
- (B) Aerial photograph of the picture of a monkey as part of the Nazca Lines geoglyphs, Peru, about 700 – 200 B.C.;
- (C) Shen Zhou (1427-1509 A.D.): Poet on a mountain top, ink on paper, China;
- (D) Line drawing by 7-year old I. Lleras (2010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

WE KNOW EDGES ARE SPECIAL FROM HUMAN (MAMMALIAN) VISION STUDIES

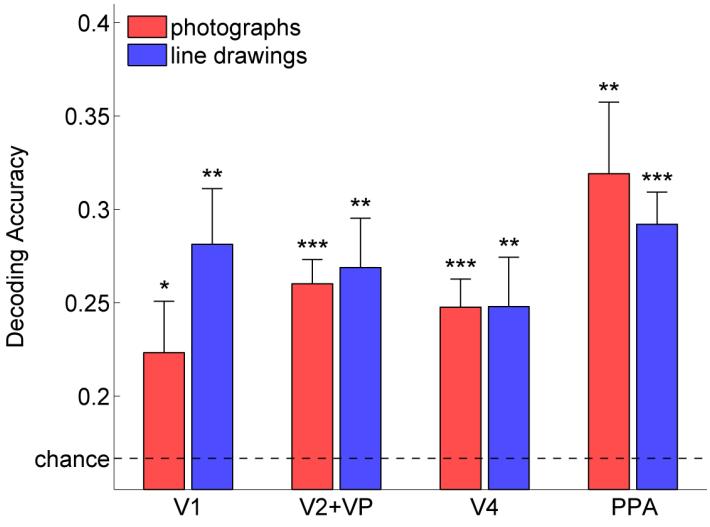
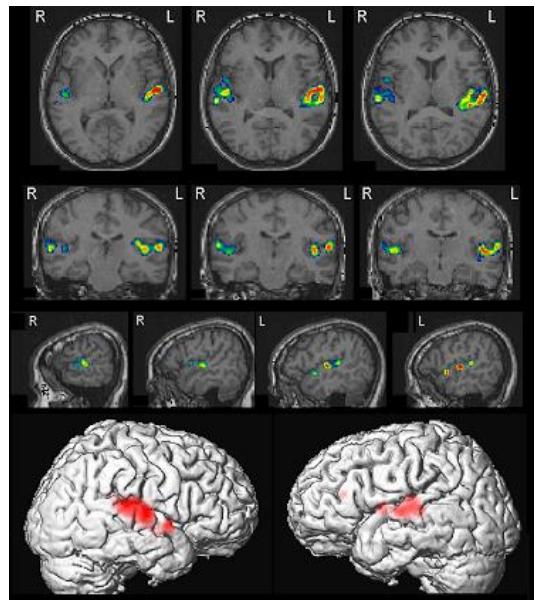


Hubel & Wiesel, 1960s





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



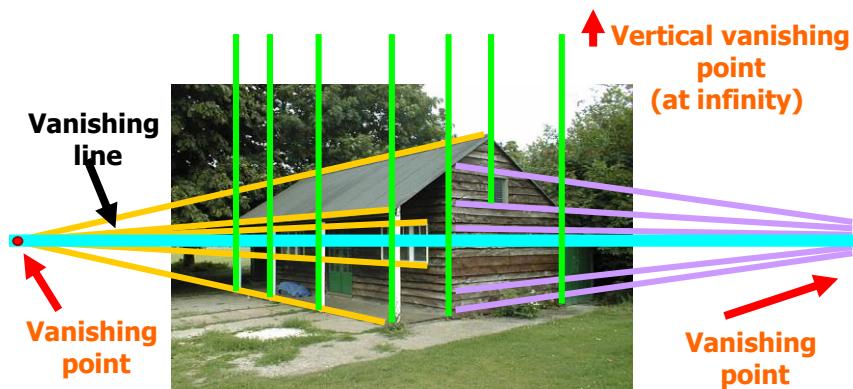
- **Goal:** Identify sudden changes (discontinuities) in an image
 - Intuitively, most semantic and shape information from the image can be encoded in the edges
 - More compact than pixels
- **Ideal:** artist's line drawing (but artist is also using object-level knowledge)



WHY DO WE CARE ABOUT EDGES?

Extract information, recognize objects

Recover geometry and viewpoint



ORIGINS OF EDGES



surface normal discontinuity

depth discontinuity

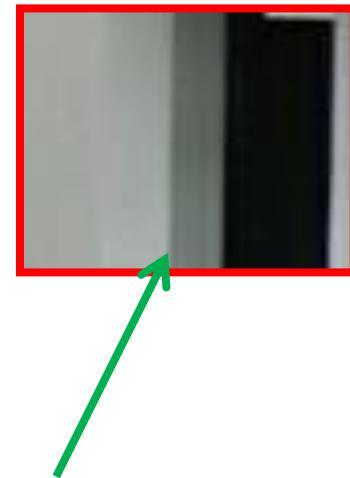
surface color discontinuity

illumination discontinuity

CLOSEUP OF EDGES



Surface normal discontinuity



CLOSEUP OF EDGES



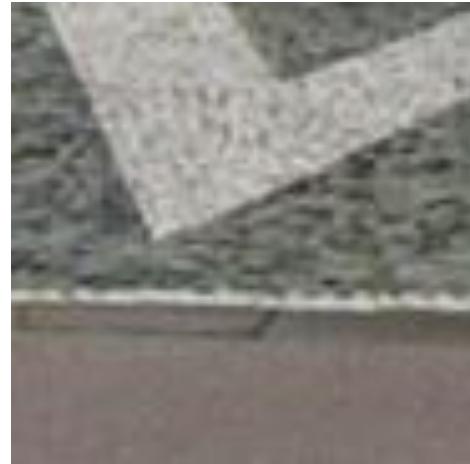
Depth discontinuity



CLOSEUP OF EDGES



Surface color discontinuity



WHAT WE WILL LEARN TODAY

1. Edge detection
2. Image Gradients
3. A simple edge detector
4. Sobel edge detector
5. Canny edge detector

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

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

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

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

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

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

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

$$\frac{df}{dx} = \frac{f(x) - f(x-1)}{1} = f'(x)$$

$$\frac{df}{dx} = f(x) - f(x-1) = f'(x)$$

$$\frac{df}{dx} = \lim_{\Delta x \rightarrow 0} \frac{f(x) - f(x - \Delta x)}{\Delta x} = f'(x)$$

Backward

$$\frac{df}{dx} = f(x) - f(x-1) = f'(x)$$

Forward

$$\frac{df}{dx} = f(x) - f(x+1) = f'(x)$$

Central

$$\frac{df}{dx} = f(x+1) - f(x-1) = f'(x)$$

Backward filter: [0 1 -1]

$$f(x) - f(x-1) = f'(x)$$

Forward: [-1 1 0]

$$f(x) - f(x+1) = f'(x)$$

Central: [1 0 -1]

$$f(x+1) - f(x-1) = f'(x)$$

Given function

$$f(x, y)$$

Gradient vector

$$\nabla f(x, y) = \begin{bmatrix} \frac{\partial f(x, y)}{\partial x} \\ \frac{\partial f(x, y)}{\partial y} \end{bmatrix} = \begin{bmatrix} f_x \\ f_y \end{bmatrix}$$

Gradient magnitude

$$|\nabla f(x, y)| = \sqrt{f_x^2 + f_y^2}$$

Gradient direction

$$\theta = \tan^{-1} \frac{f_x}{f_y}$$

Given function

$$f(x, y)$$

Gradient vector

$$\nabla f(x, y) = \begin{bmatrix} \frac{\partial f(x, y)}{\partial x} \\ \frac{\partial f(x, y)}{\partial y} \end{bmatrix} = \begin{bmatrix} f_x \\ f_y \end{bmatrix}$$

Gradient magnitude

$$|\nabla f(x, y)| = \sqrt{f_x^2 + f_y^2}$$

Gradient direction

$$\theta = \tan^{-1} \frac{f_x}{f_y} \quad \theta = \tan^{-1} \left(\frac{\partial f}{\partial y} / \frac{\partial f}{\partial x} \right)$$

What does this filter do?

$$\frac{1}{3} \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}$$

What about this filter?

$$\frac{1}{3} \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}$$

$$\frac{1}{3} \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix}$$

$$I = \begin{bmatrix} 10 & 10 & 20 & 20 & 20 \\ 10 & 10 & 20 & 20 & 20 \\ 10 & 10 & 20 & 20 & 20 \\ 10 & 10 & 20 & 20 & 20 \\ 10 & 10 & 20 & 20 & 20 \end{bmatrix}$$

What happens when we apply this filter?

$$I = \begin{bmatrix} 10 & 10 & 20 & 20 & 20 \\ 10 & 10 & 20 & 20 & 20 \\ 10 & 10 & 20 & 20 & 20 \\ 10 & 10 & 20 & 20 & 20 \\ 10 & 10 & 20 & 20 & 20 \end{bmatrix}$$

$$\frac{1}{3} \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix}$$

What happens when we apply this filter?

$$I = \begin{bmatrix} 10 & 10 & 20 & 20 & 20 \\ 10 & 10 & 20 & 20 & 20 \\ 10 & 10 & 20 & 20 & 20 \\ 10 & 10 & 20 & 20 & 20 \\ 10 & 10 & 20 & 20 & 20 \end{bmatrix}$$

$$\frac{1}{3} \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix}$$

$$I_y = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Now let's try the other filter!

$$I = \begin{bmatrix} 10 & 10 & 20 & 20 & 20 \\ 10 & 10 & 20 & 20 & 20 \\ 10 & 10 & 20 & 20 & 20 \\ 10 & 10 & 20 & 20 & 20 \\ 10 & 10 & 20 & 20 & 20 \end{bmatrix}$$

$$\frac{1}{3} \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}$$

What happens when we apply this filter?

$$I = \begin{bmatrix} 10 & 10 & 20 & 20 & 20 \\ 10 & 10 & 20 & 20 & 20 \\ 10 & 10 & 20 & 20 & 20 \\ 10 & 10 & 20 & 20 & 20 \\ 10 & 10 & 20 & 20 & 20 \end{bmatrix}$$

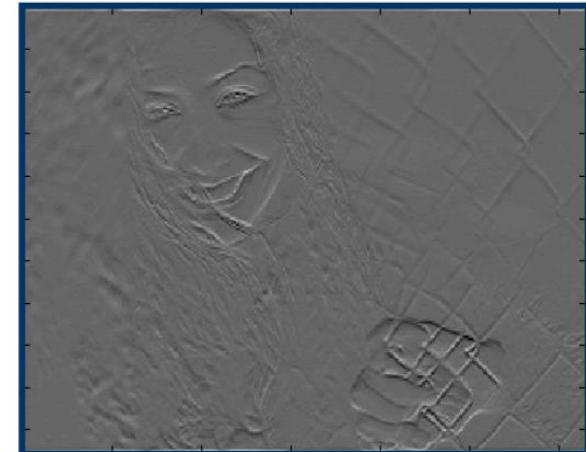
$$\frac{1}{3} \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}$$

$$I_x = \begin{bmatrix} 0 & 10 & 10 & 0 & 0 \\ 0 & 10 & 10 & 0 & 0 \\ 0 & 10 & 10 & 0 & 0 \\ 0 & 10 & 10 & 0 & 0 \\ 0 & 10 & 10 & 0 & 0 \end{bmatrix}$$

3X3 IMAGE GRADIENT FILTERS

$$\frac{1}{3} \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}$$

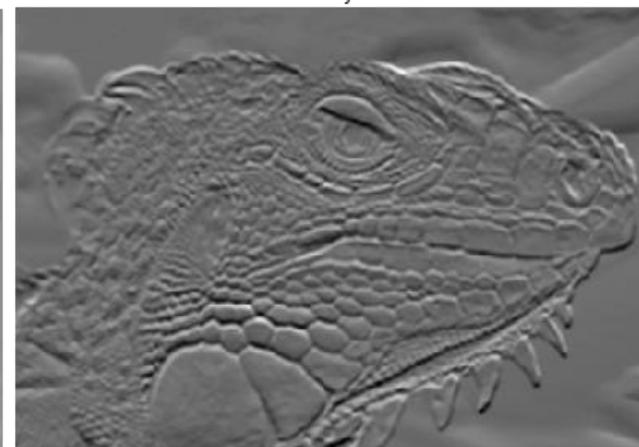
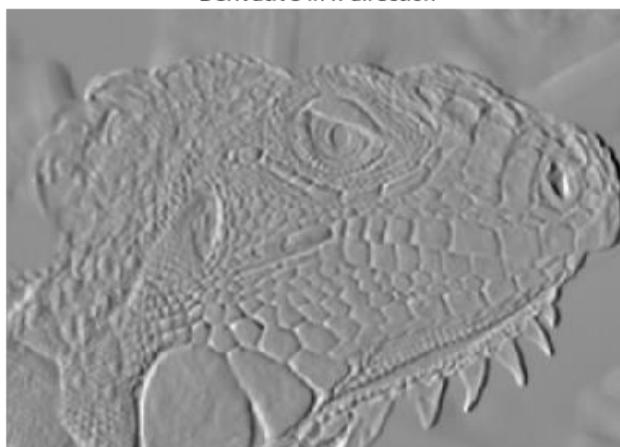
$$\frac{1}{3} \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix}$$



3X3 IMAGE GRADIENT FILTERS

$$\frac{1}{3} \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}$$

$$\frac{1}{3} \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix}$$

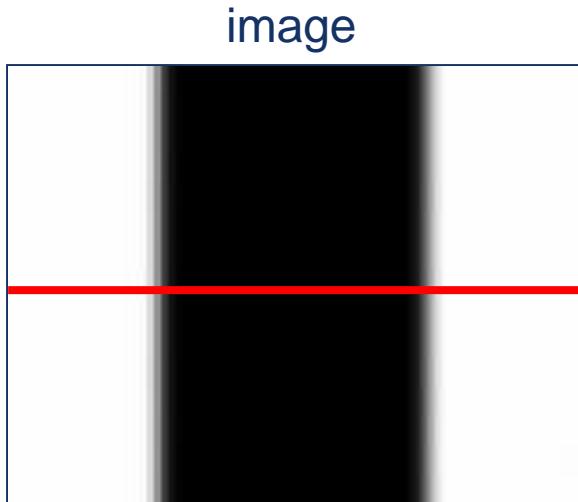


WHAT WE WILL LEARN TODAY

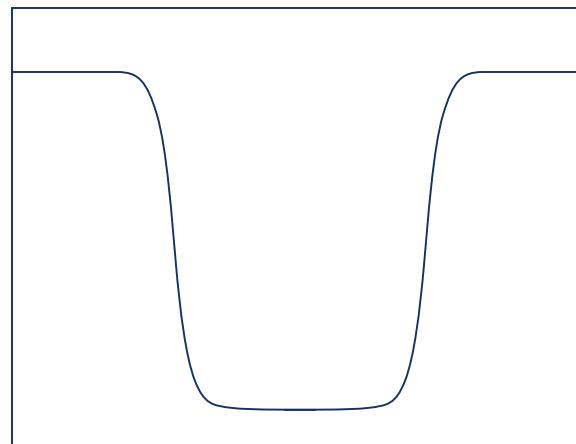
1. Edge detection
2. Image Gradients
3. A simple edge detector
4. Sobel edge detector
5. Canny edge detector

CHARACTERIZING EDGES

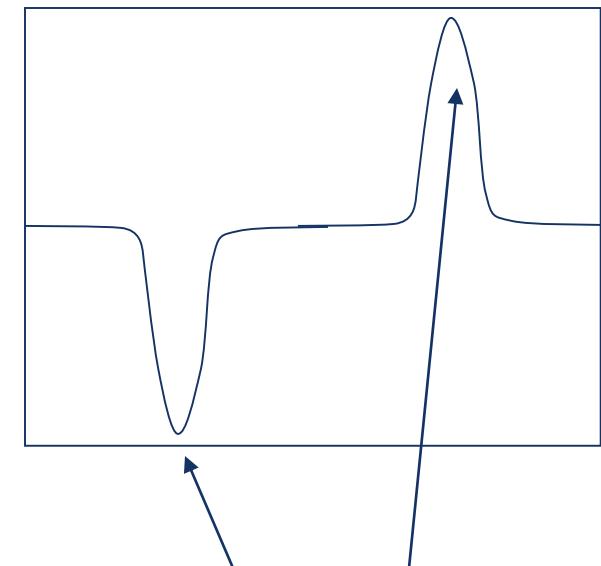
- An edge is a place of rapid change in the image intensity function



intensity function
(along horizontal scanline)



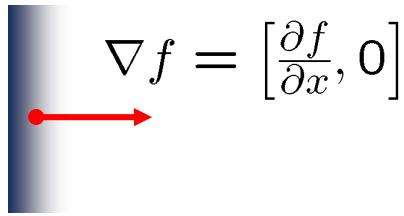
first derivative

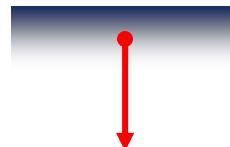


edges correspond to
extrema of derivative

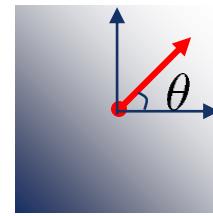
$$\nabla f = \left[\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y} \right]$$

1. The gradient of an image:


$$\nabla f = \left[\frac{\partial f}{\partial x}, 0 \right]$$



$$\nabla f = \left[0, \frac{\partial f}{\partial y} \right]$$


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

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

The gradient direction is given by $\theta = \tan^{-1} \left(\frac{\partial f}{\partial y} / \frac{\partial f}{\partial x} \right)$

The *edge strength* is given by the gradient magnitude

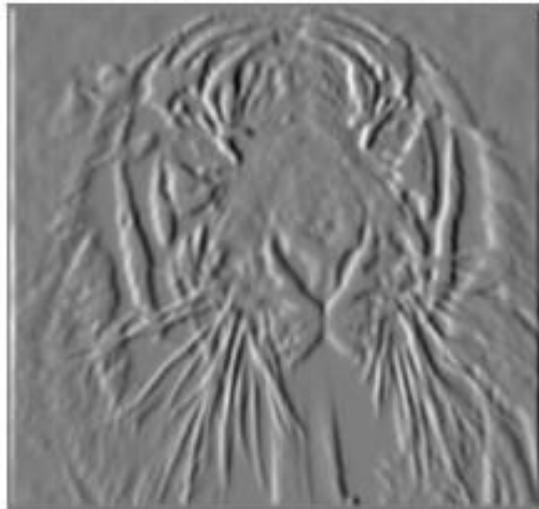
$$\|\nabla f\| = \sqrt{\left(\frac{\partial f}{\partial x} \right)^2 + \left(\frac{\partial f}{\partial y} \right)^2}$$

FINITE DIFFERENCES: EXAMPLE

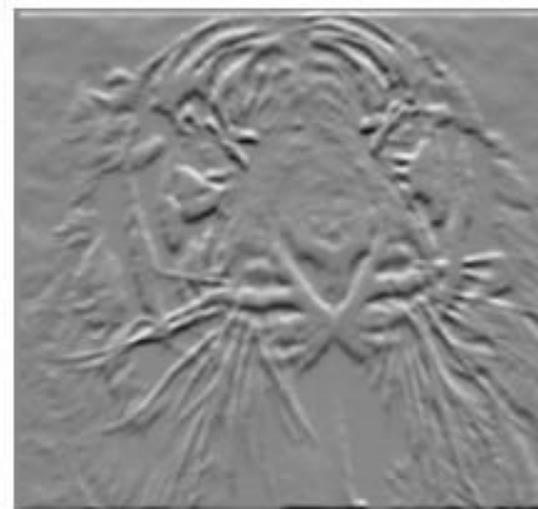
Original
Image



x-direction

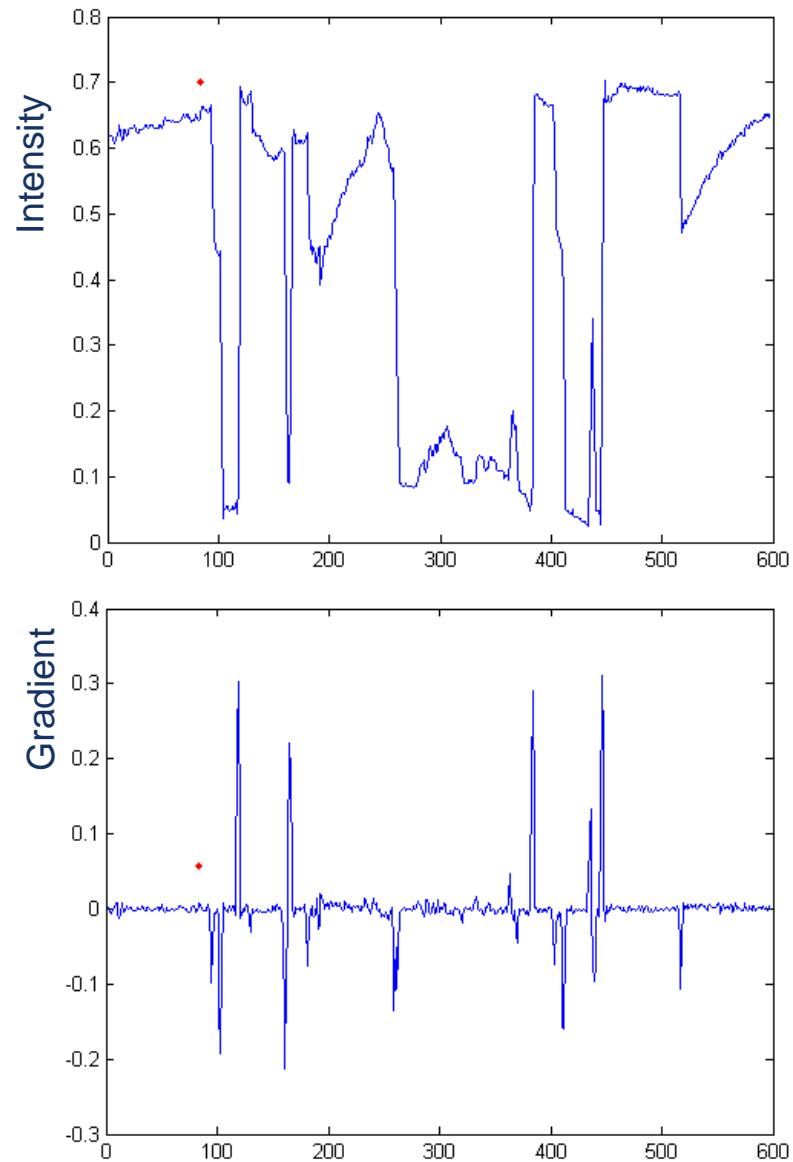
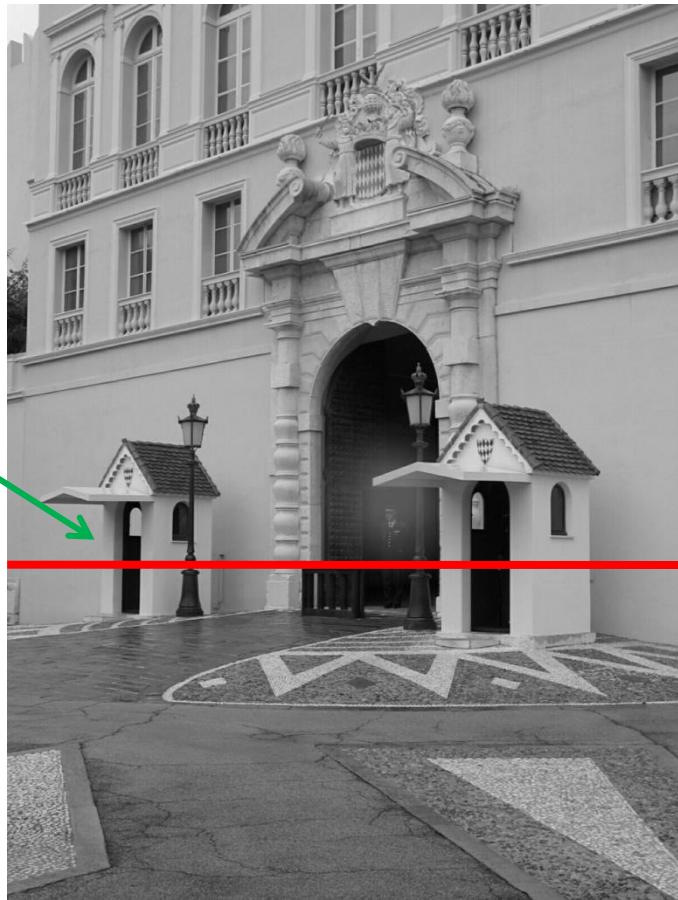


Gradient
magnitude



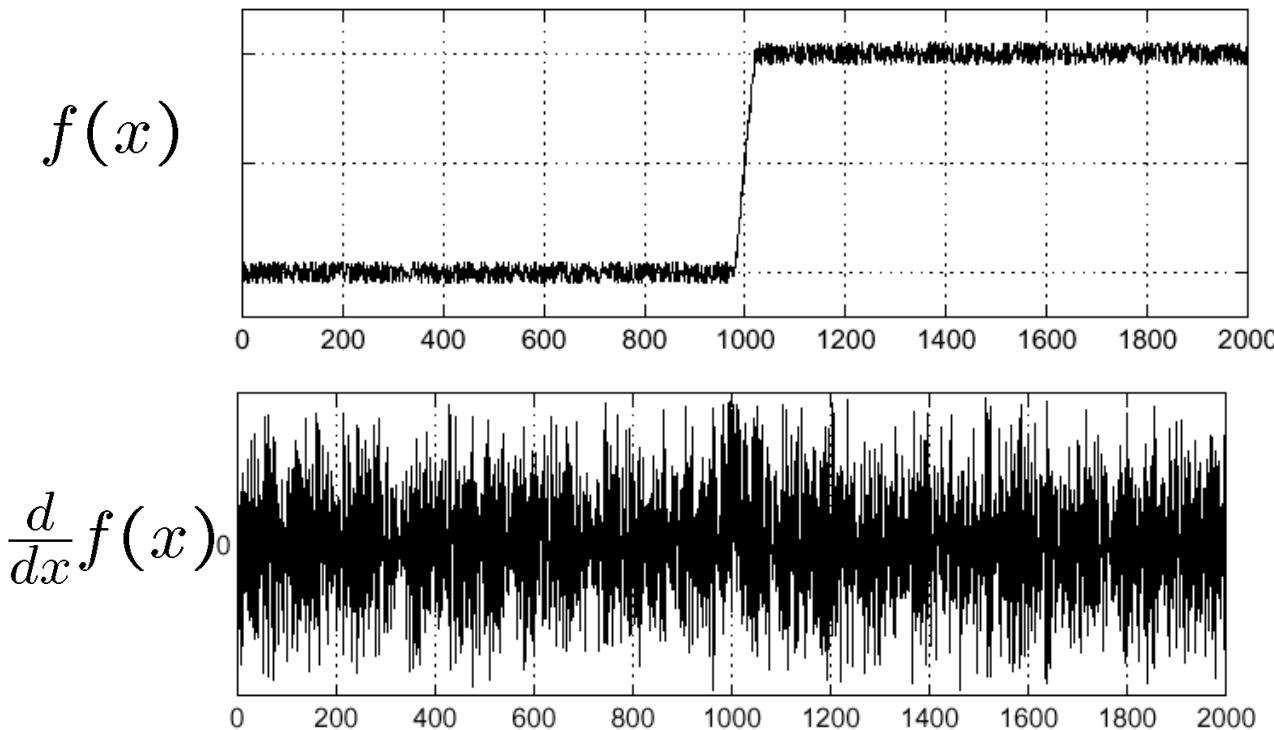
y-direction

INTENSITY PROFILE



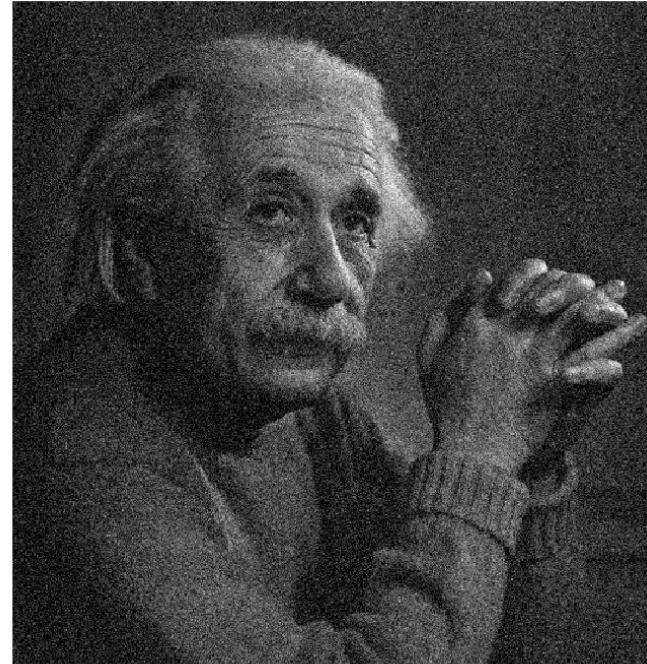
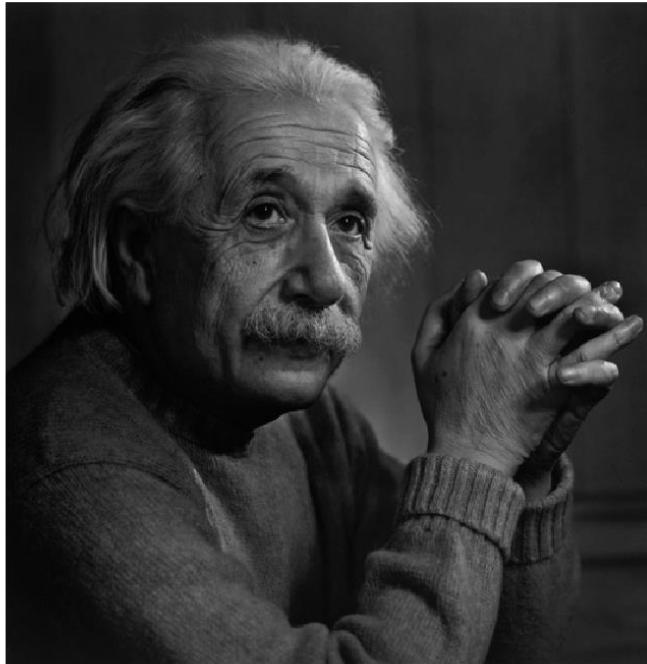
Consider a single row or column of the image

- Plotting intensity as a function of position gives a signal



Where is the edge?

EFFECTS OF NOISE



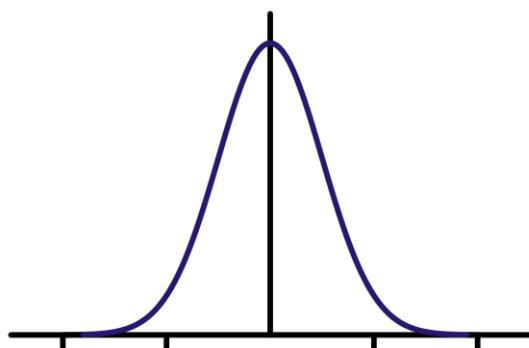
- Finite difference filters respond strongly to noise
 - Image noise results in pixels that look very different from their neighbors
 - Generally, the larger the noise the stronger the response
- What is to be done?
 - Smoothing the image should help, by forcing pixels different to their neighbors (=noise pixels?) to look more like neighbors

SMOOTHING WITH DIFFERENT FILTERS

1. Mean smoothing

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

2. Gaussian (smoothing * derivative)



$$\begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} \quad [1 \quad 2 \quad 1]$$

SMOOTHING WITH DIFFERENT FILTERS

Mean



Gaussian



Median



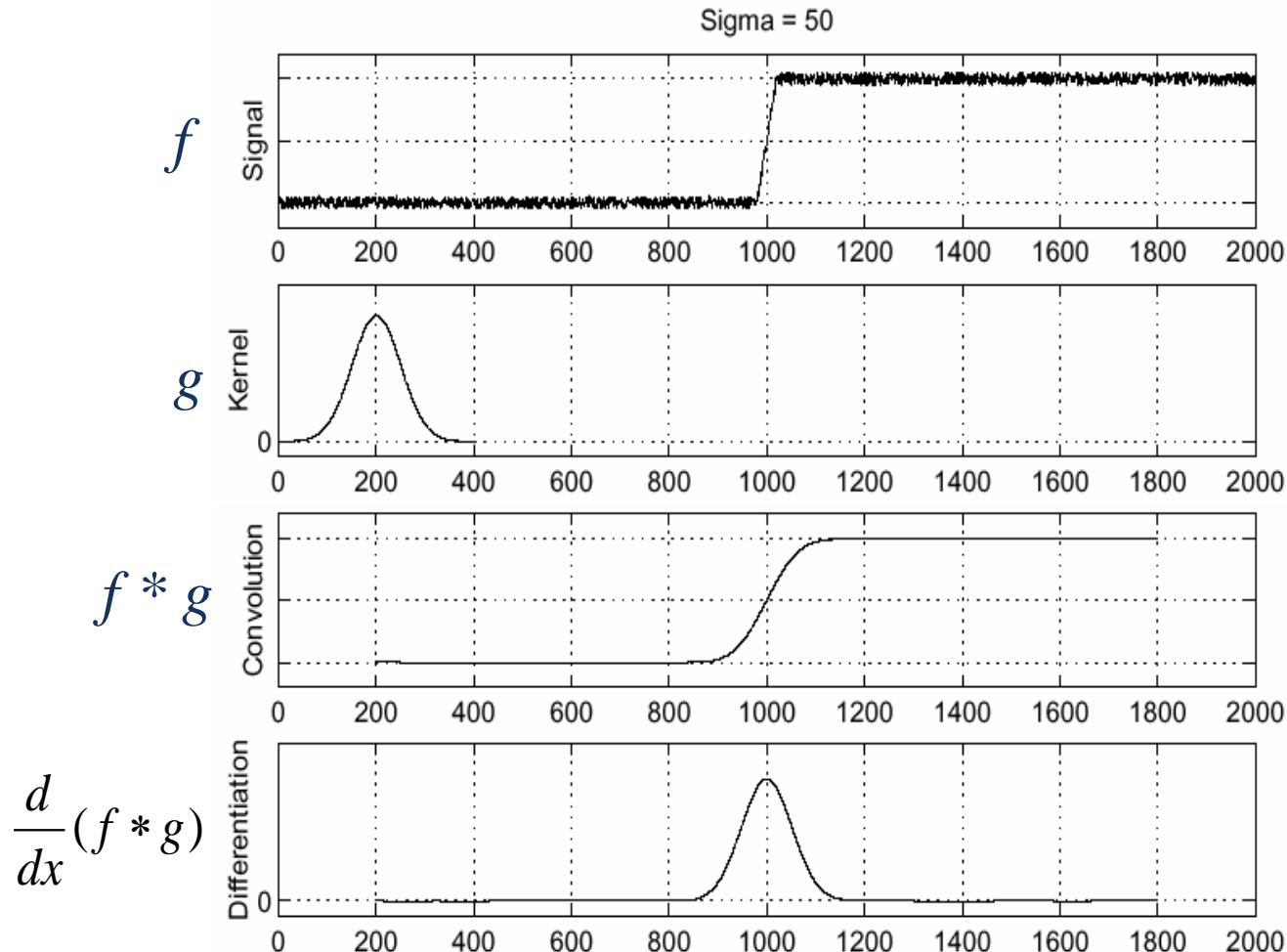
3x3

5x5

7x7



SOLUTION: SMOOTH FIRST

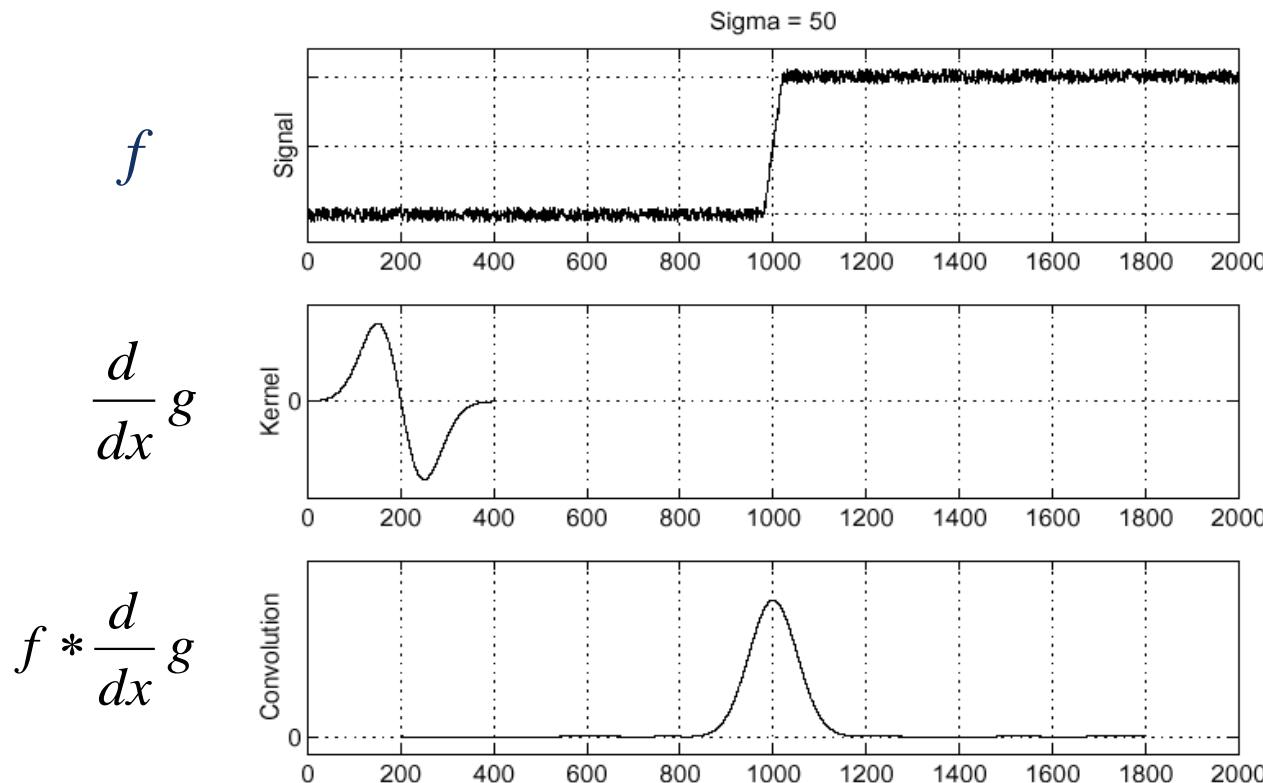


to find edges, look for peaks in $\frac{d}{dx}(f * g)$

DERIVATIVE THEOREM OF CONVOLUTION

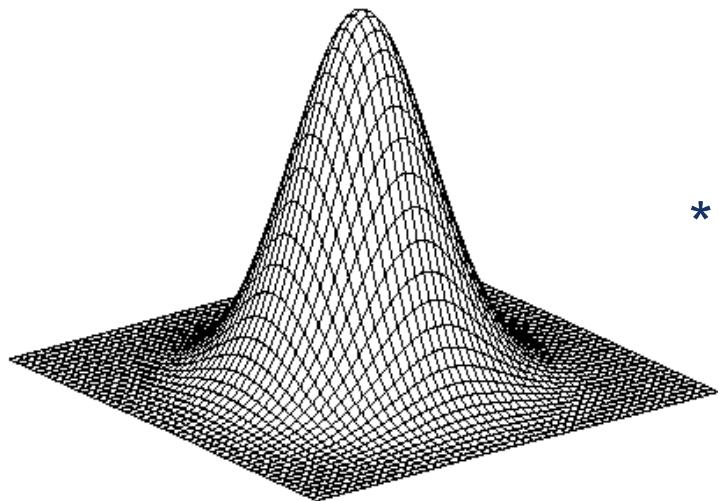
- This theorem gives us a very useful property:
- This saves us one operation:

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

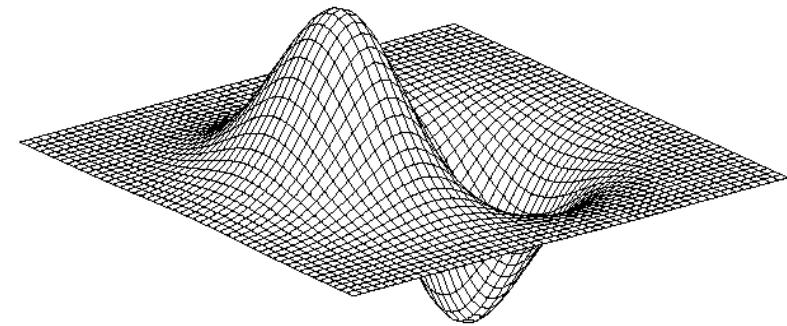


Source: S. Seitz

DERIVATIVE OF GAUSSIAN FILTER



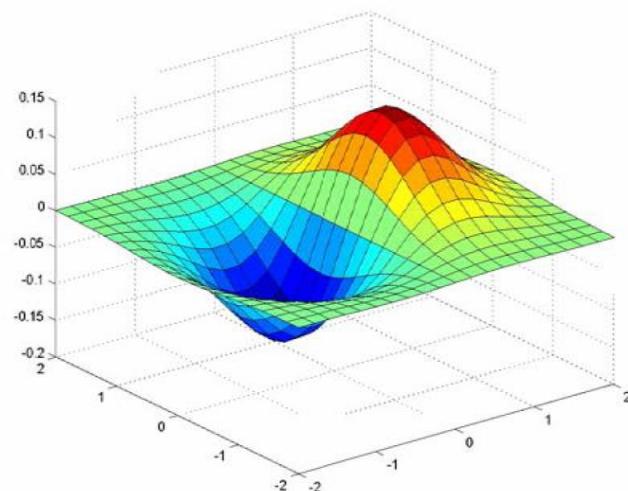
$$* [1 \ 0 \ -1] =$$



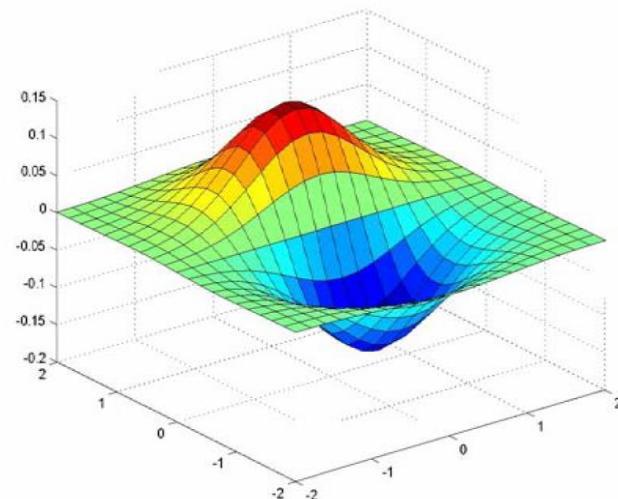
2D-Gaussian

x - derivative

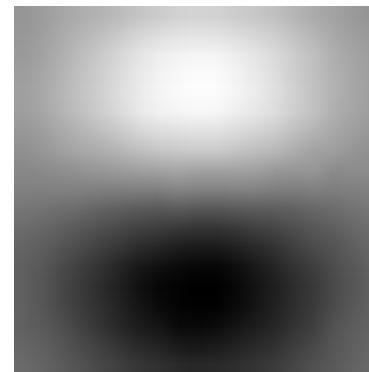
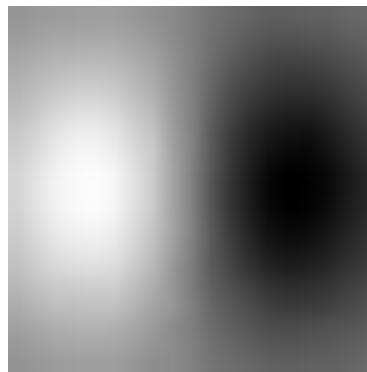
DERIVATIVE OF GAUSSIAN FILTER



x-direction



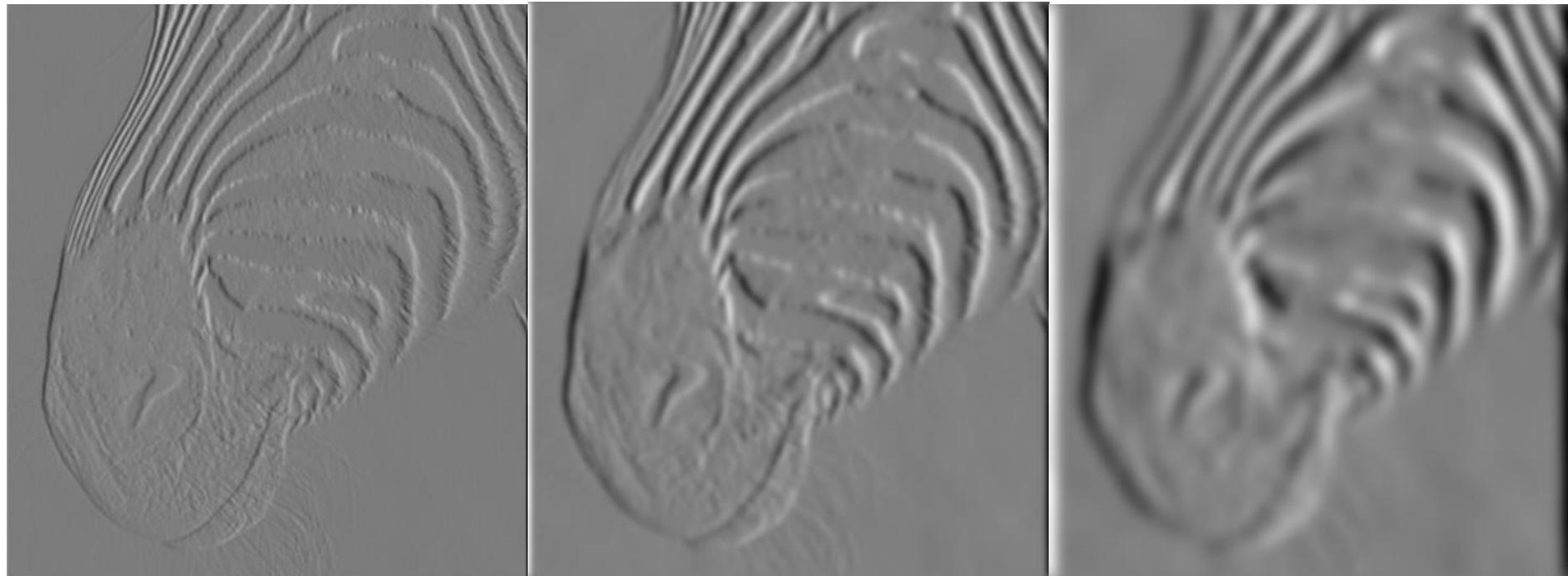
y-direction



DERIVATIVE OF GAUSSIAN FILTER



TRADEOFF BETWEEN SMOOTHING AT DIFFERENT SCALES



1 pixel

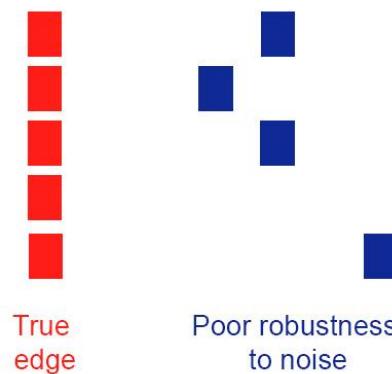
3 pixels

7 pixels

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

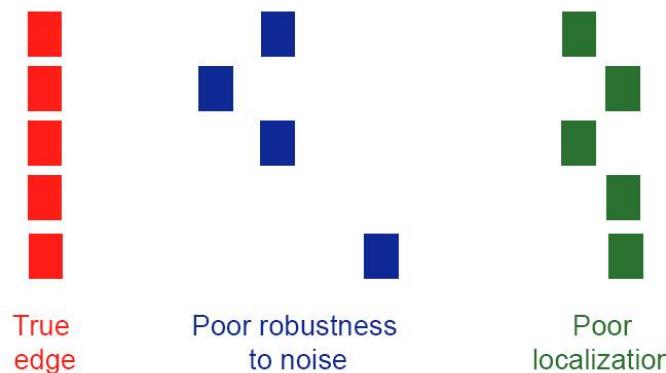
Criteria for an “optimal” edge detector:

- **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)



Criteria for an “optimal” edge detector:

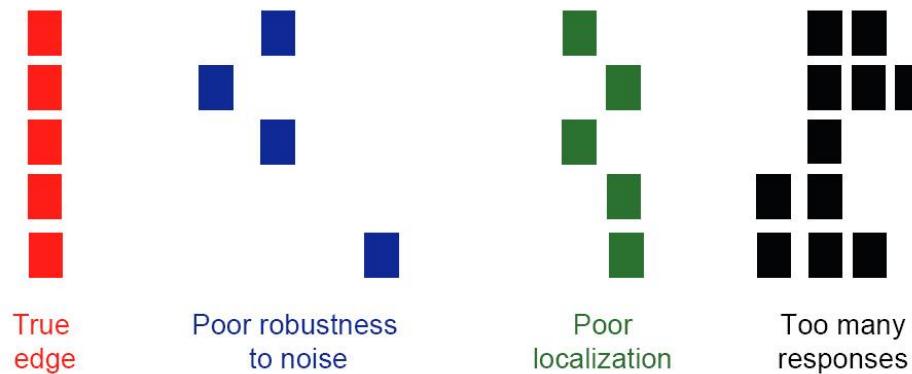
- **Good detection:** the optimal detector must minimize the probability of false positives (detecting spurious edges caused by noise), as well as that of false negatives (missing real edges)
- **Good localization:** the edges detected must be as close as possible to the true edges



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

1. Edge detection
2. Image Gradients
3. A simple edge detector
4. Sobel Edge detector
5. Canny edge detector

1. uses two 3×3 kernels which are convolved with the original image to calculate approximations of the derivatives
2. 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

1. 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

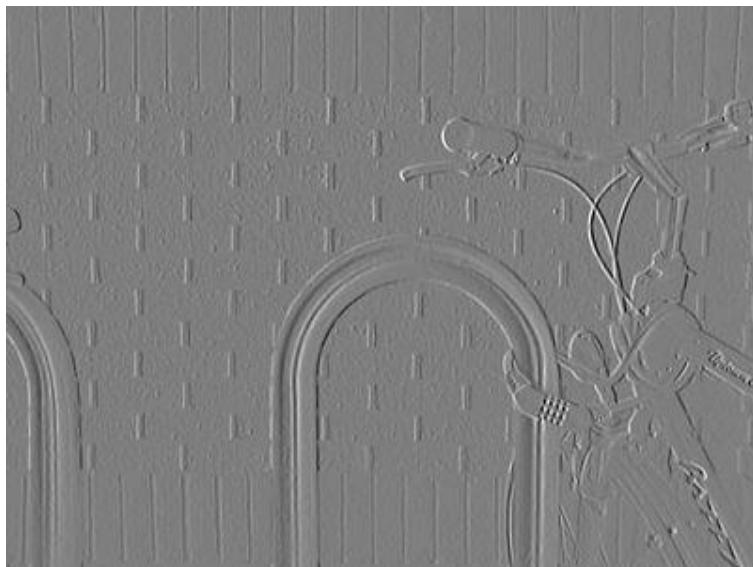
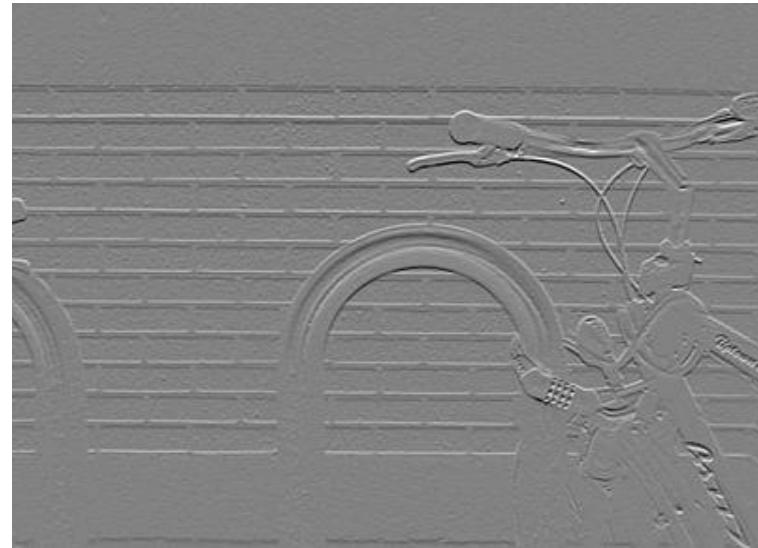
1. Magnitude:

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

2. Angle or direction of the gradient:

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

SOBEL FILTER EXAMPLE



WHAT WE WILL LEARN TODAY

1. Edge detection
2. Image Gradients
3. A simple edge detector
4. Sobel Edge detector
5. Canny edge detector

- This is probably the most widely used edge detector in computer vision
- Theoretical model: step-edges corrupted by additive Gaussian noise
- Canny has shown that the first derivative of the Gaussian closely approximates the operator that optimizes the product of *signal-to-noise ratio* and localization

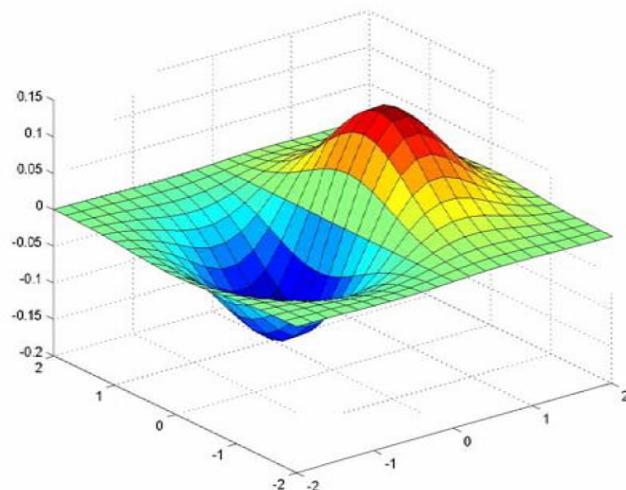
J. Canny, ***A Computational Approach To Edge Detection***, IEEE Trans. Pattern Analysis and Machine Intelligence, 8:679-714, 1986.

- 1) Suppress Noise
- 2) Compute gradient magnitude and direction
- 3) Apply Non-Maximum Suppression
 - Assures minimal response
- 4) Use hysteresis and connectivity analysis to detect edges

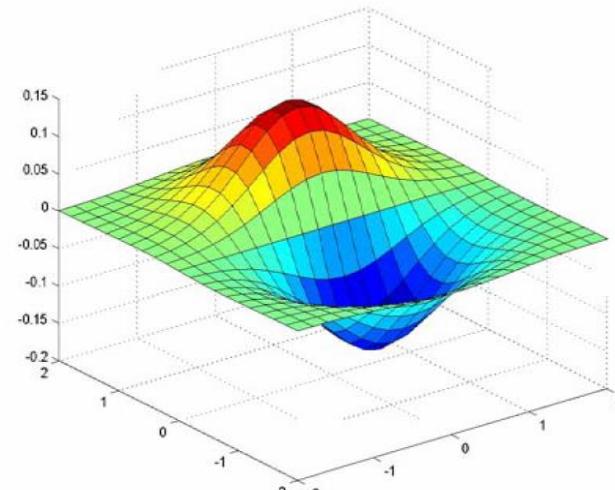
EXAMPLE



DERIVATIVE OF GAUSSIAN FILTER

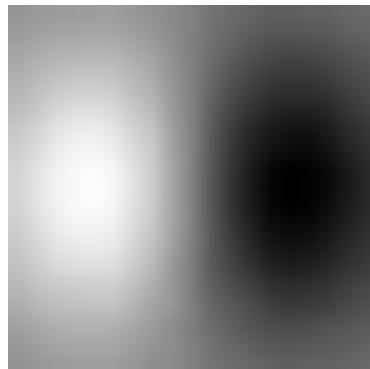


x-direction

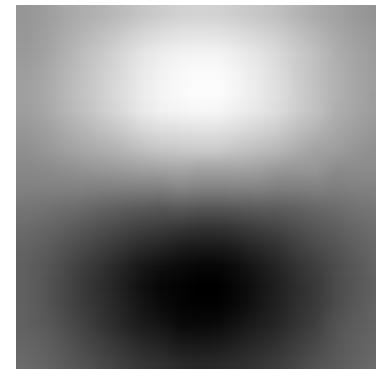


y-direction

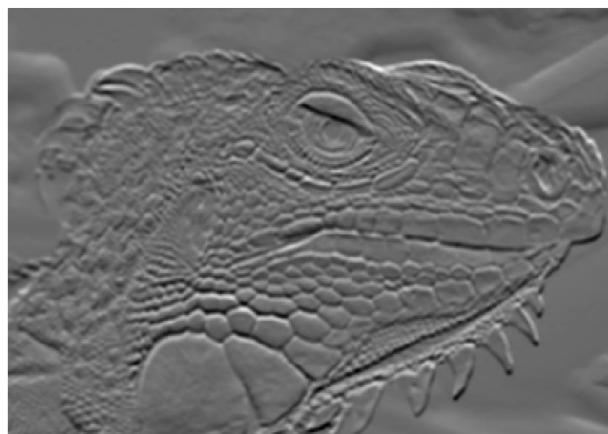
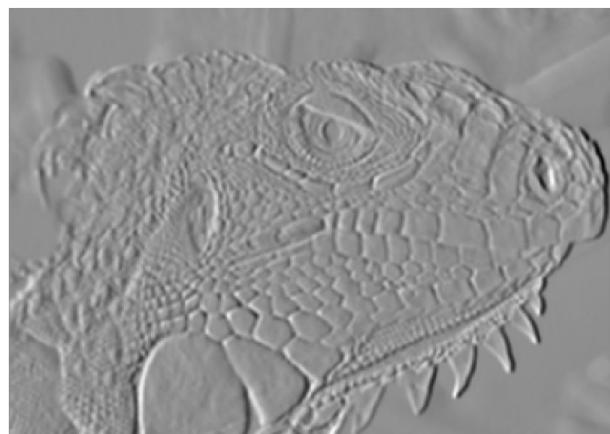
$$\begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix}$$



$$\begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$



COMPUTE GRADIENTS



X-Derivative of Gaussian



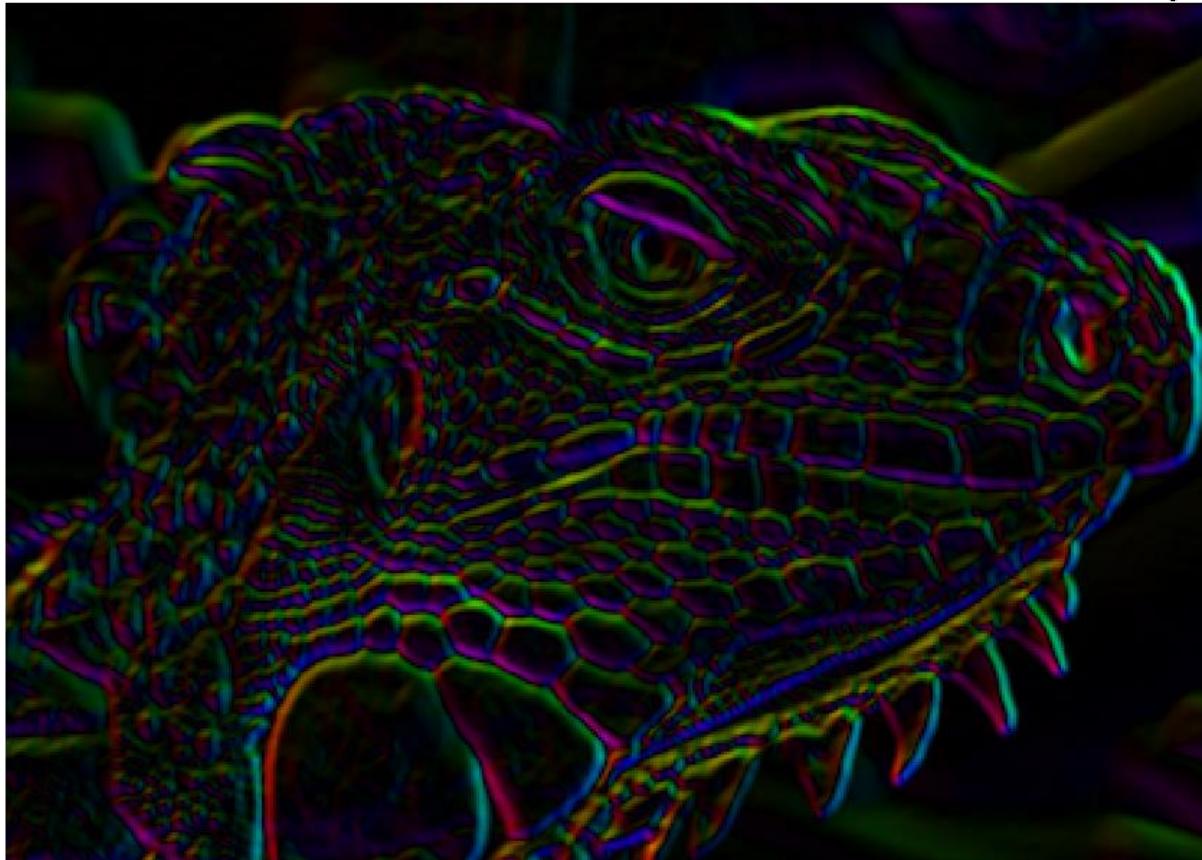
Y-Derivative of Gaussian

Gradient Magnitude

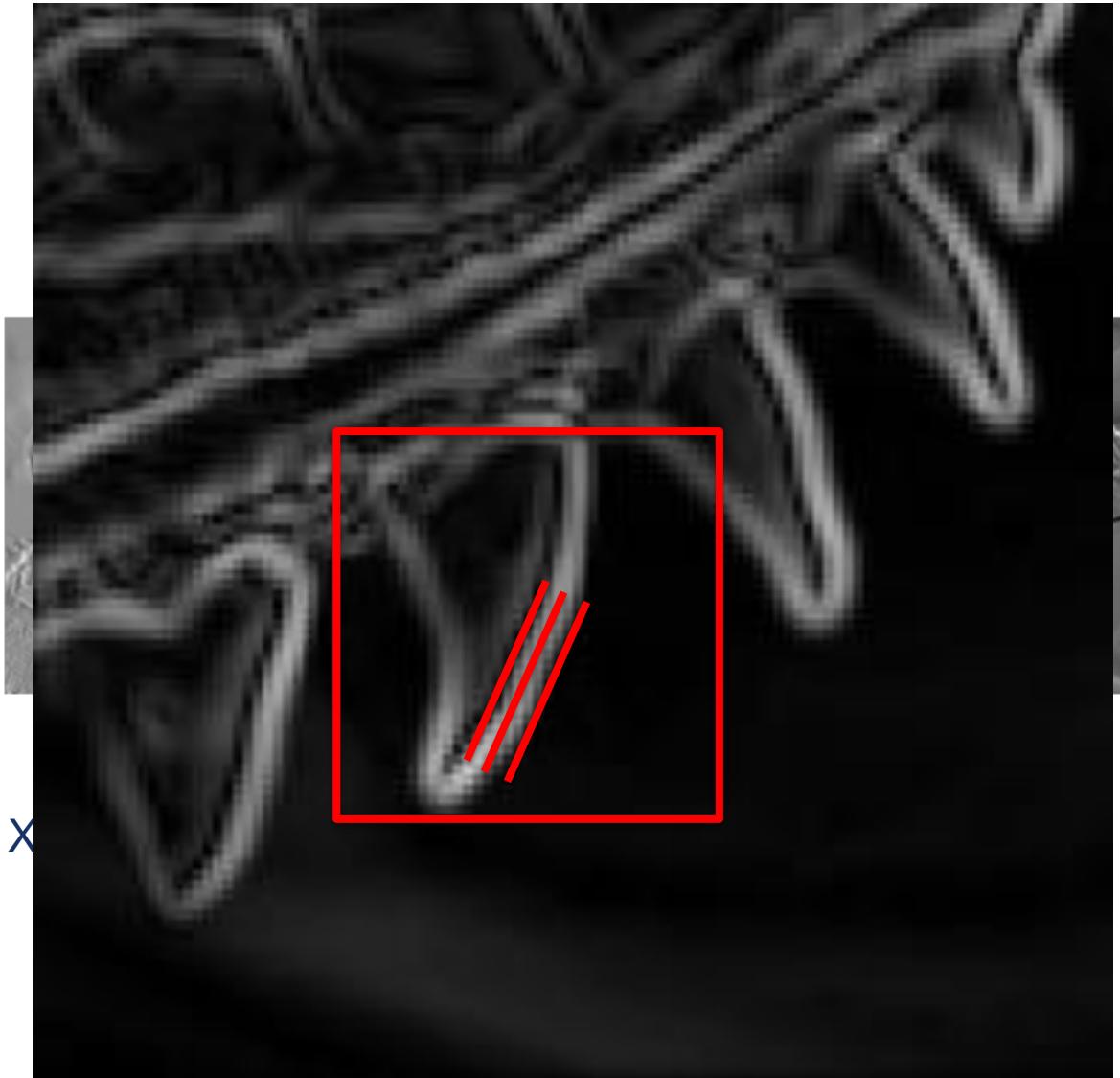
Source: J. Hayes

GET ORIENTATION AT EACH PIXEL

$$\theta = \tan^{-1} \left(\frac{\partial f}{\partial y} / \frac{\partial f}{\partial x} \right)$$



COMPUTE GRADIENTS



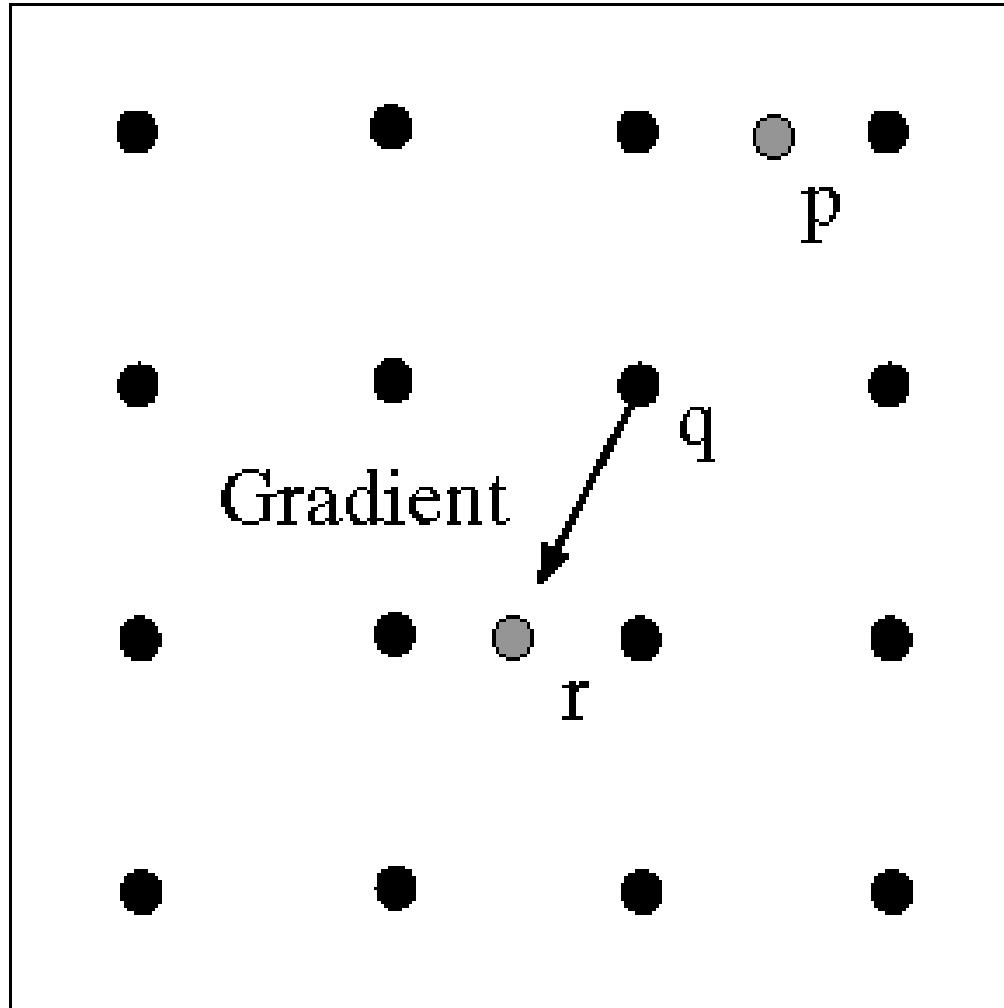
Gradient Magnitude

1. Suppress Noise
2. Compute gradient magnitude and direction
3. Apply Non-Maximum Suppression
 - Assures minimal response

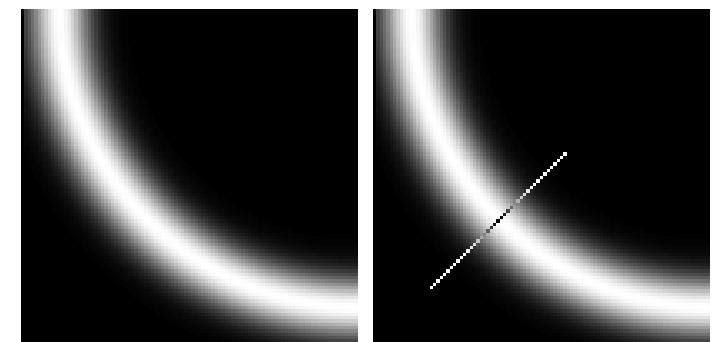
NON-MAXIMUM SUPPRESSION

1. Edge occurs where gradient reaches a maxima
2. Suppress non-maxima gradient even if it passes threshold
3. Only eight angle directions possible
 - Suppress all pixels in each direction which are not maxima
 - Do this in each marked pixel neighborhood

NON-MAXIMUM SUPPRESSION



At q , we have a maximum if the value is larger than those at both p and at r . Interpolate to get these values.



Source: D. Forsyth

NON-MAX SUPPRESSION



Before



After

1. Suppress Noise
2. Compute gradient magnitude and direction
3. Apply Non-Maximum Suppression
 - Assures minimal response
4. Use hysteresis and connectivity analysis to detect edges

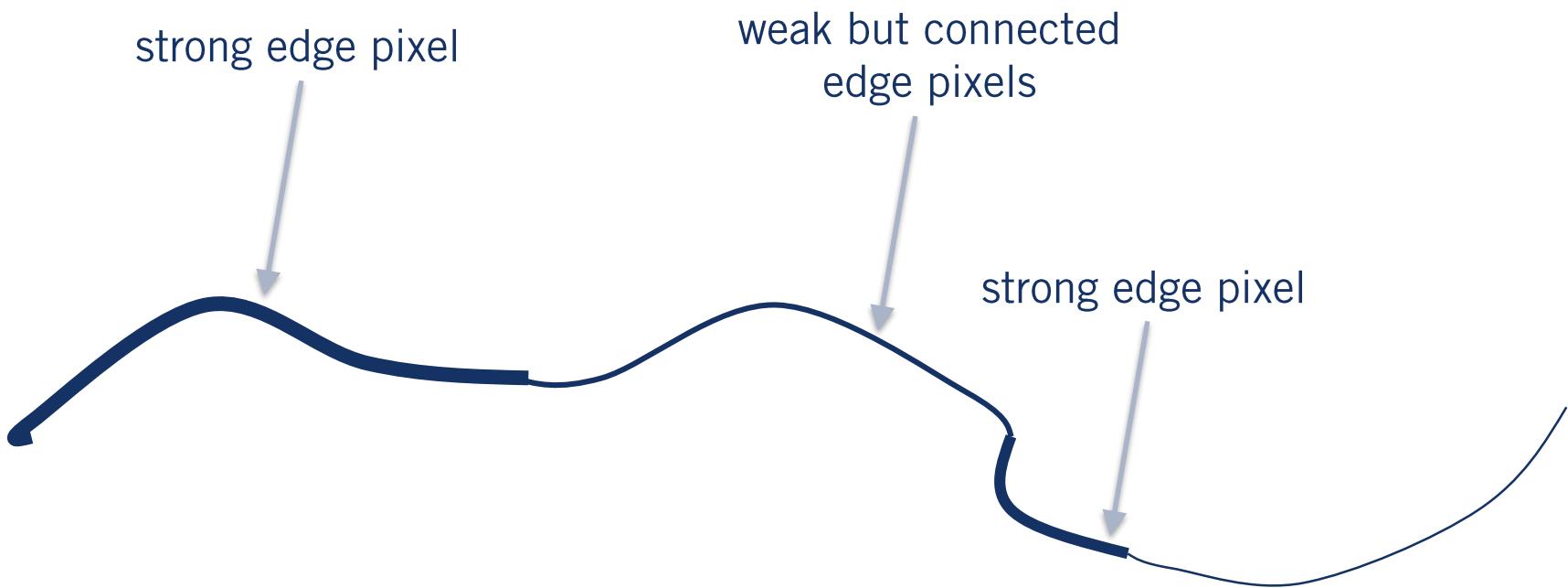
Define two thresholds: Low and High

- If less than Low, not an edge
- If greater than High, strong edge
- If between Low and High, weak edge

If the gradient at a pixel is

- 1) above High, declare it as an ‘strong edge pixel’
- 2) below Low, declare it as a “non-edge-pixel”
- 3) between Low and High
 - Consider its neighbors iteratively then declare it an “edge pixel” if it is connected to a ‘strong edge pixel’ directly or via pixels between Low and High

HYSTERESIS THRESHOLDING



Source: S. Seitz

FINAL CANNY EDGES



1. Filter image with x, y derivatives of Gaussian
2. Find magnitude and orientation of gradient
3. Non-maximum suppression:
 - Thin multi-pixel wide “ridges” down to single pixel width
4. Thresholding and linking (hysteresis):
 - Define two thresholds: low and high
 - Use the high threshold to start edge curves and the low threshold to continue them



Original Image

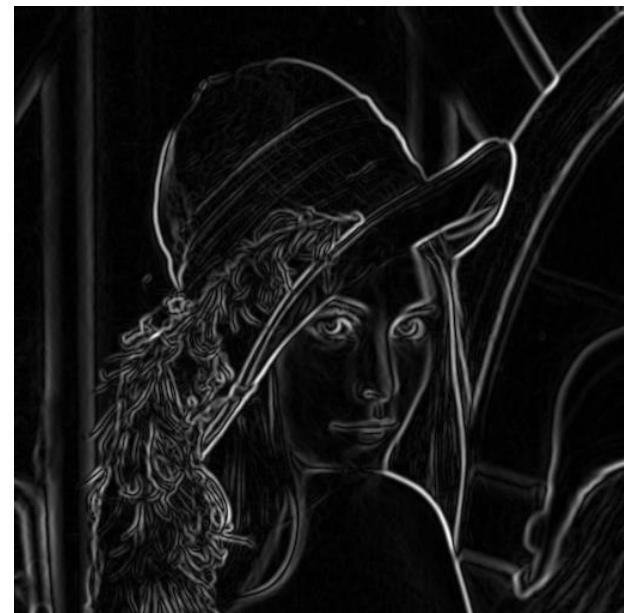
CANNY EDGE DETECTOR



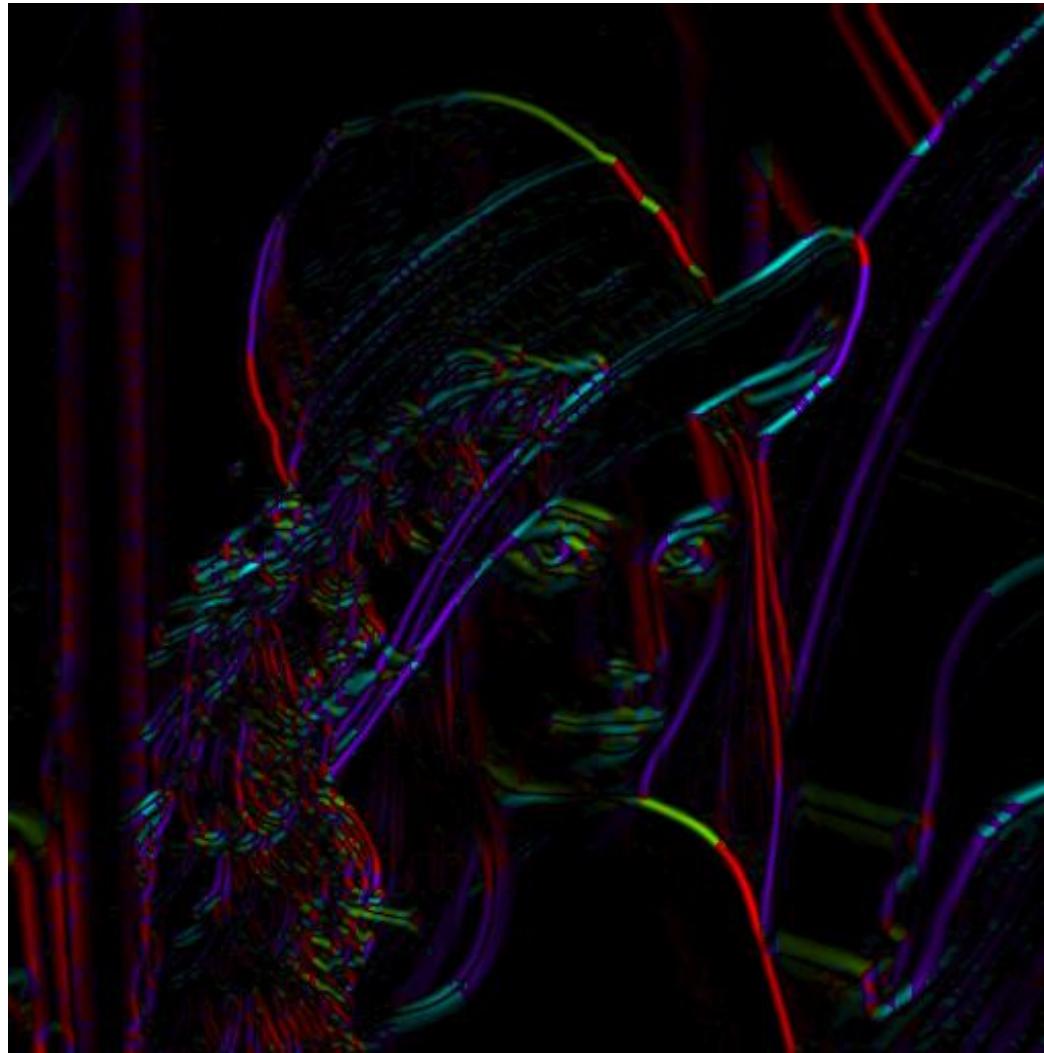
X-Derivative



Y-Derivative



Gradient Magnitude



Orientation at each pixel



Before non-max suppression



After non-max suppression



Hysteresis thresholding

CANNY EDGE DETECTOR



Final Canny Edges

EFFECT OF σ (GAUSSIAN KERNEL SPREAD/SIZE)



original



Canny with $\sigma = 1$



Canny with $\sigma = 2$

The choice of σ depends on desired behavior

- large σ detects large scale edges
- small σ detects fine features