# Lecture: Edge Detection

Juan Carlos Niebles and Ranjay Krishna
Stanford Vision and Learning Lab

# What we will learn today

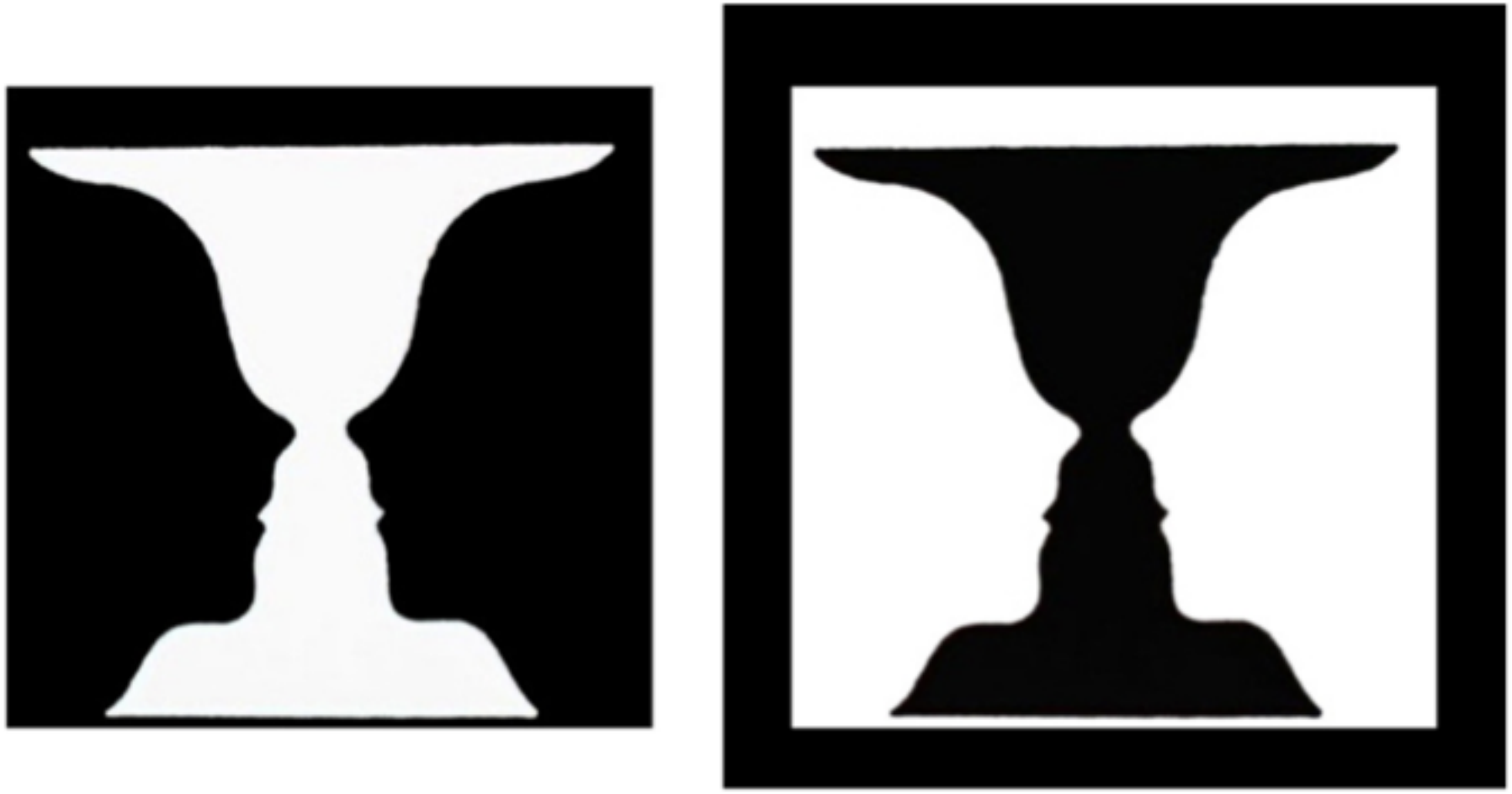
- Edge detection
- Image Gradients
- A simple edge detector
- Sobel edge detector
- Canny edge detector
- Hough Transform

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
- Canny edge detector
- Hough Transform

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

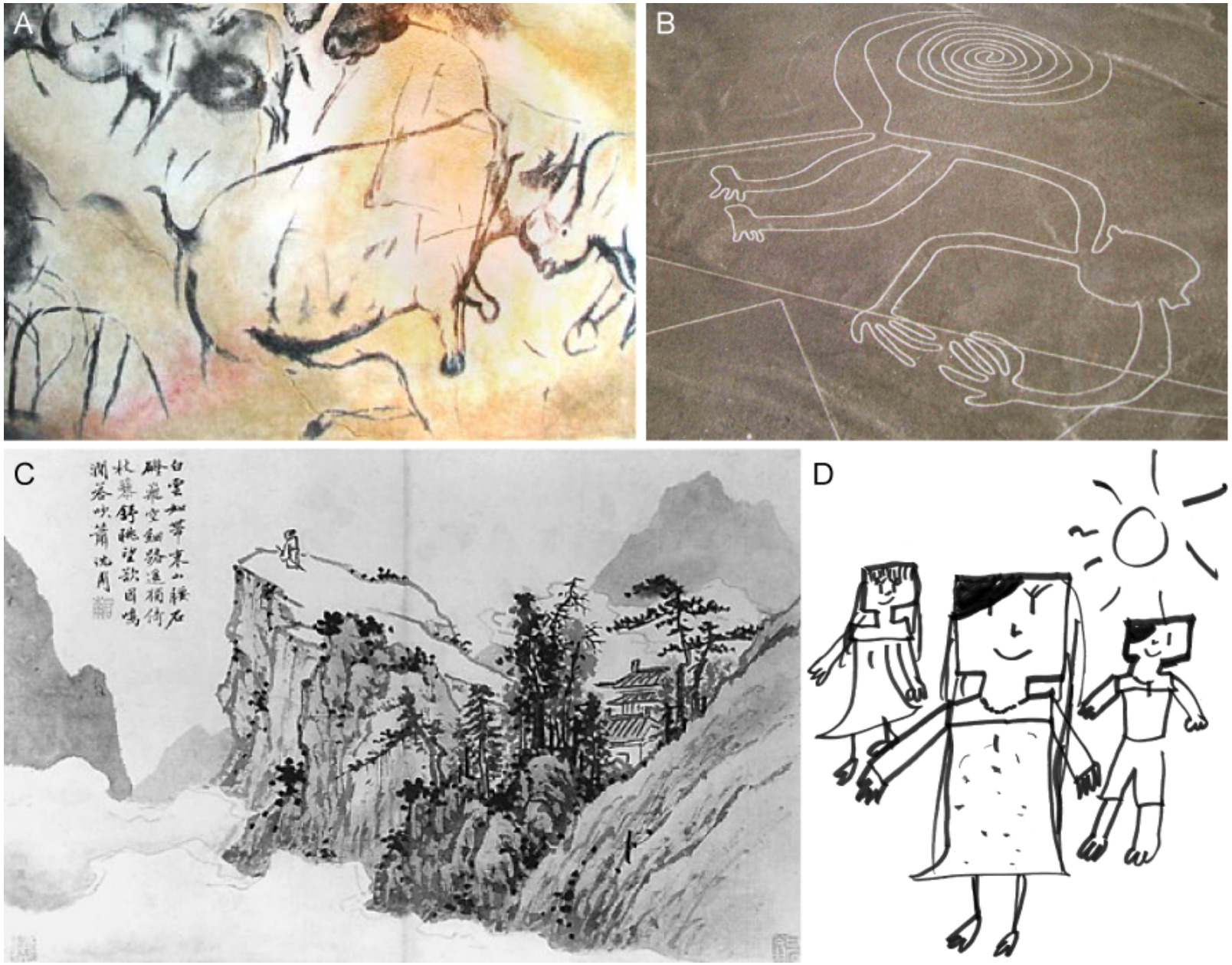


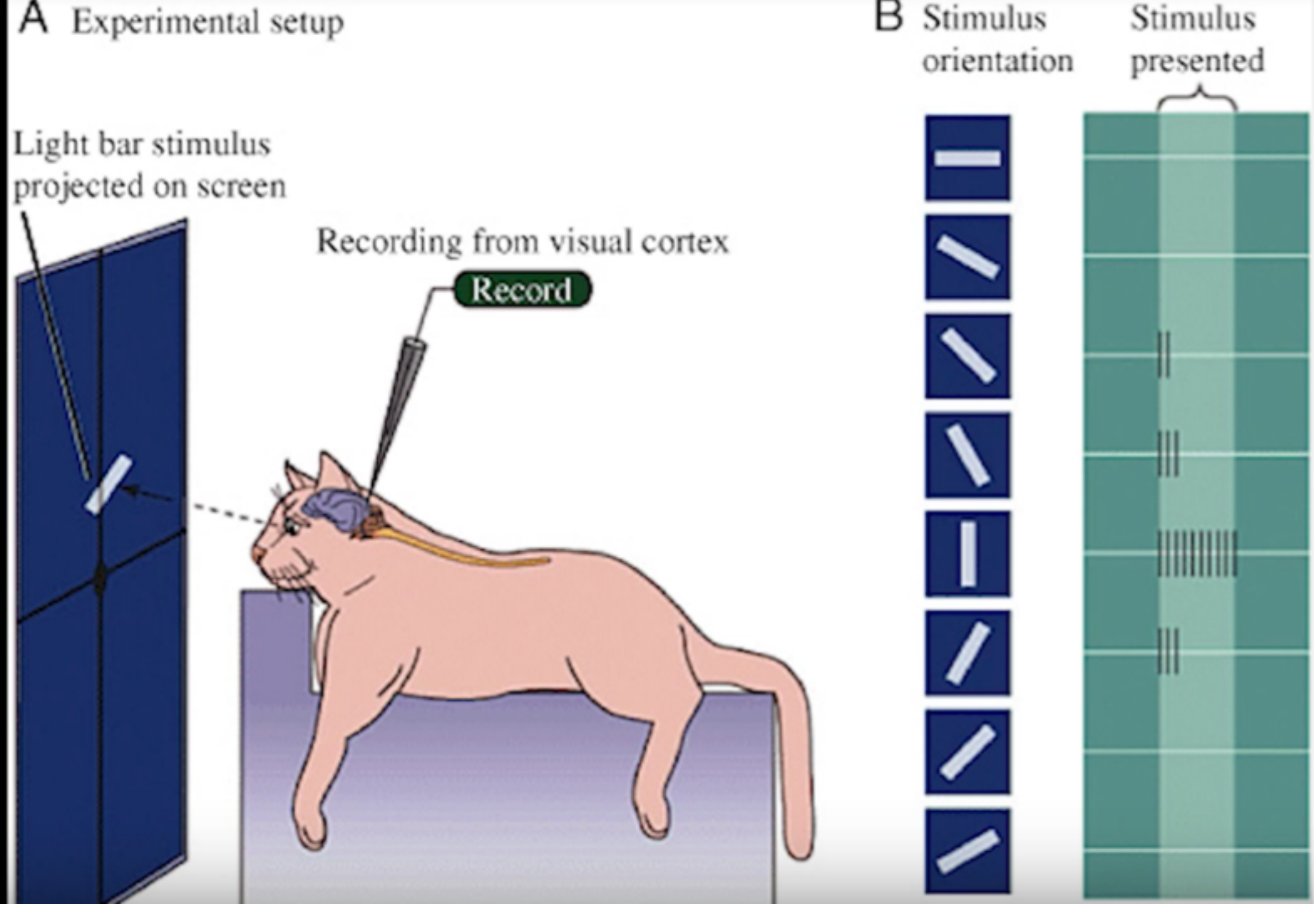
(A) Cave painting at Chauvet, France, about 30,000 B.C.;

(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;

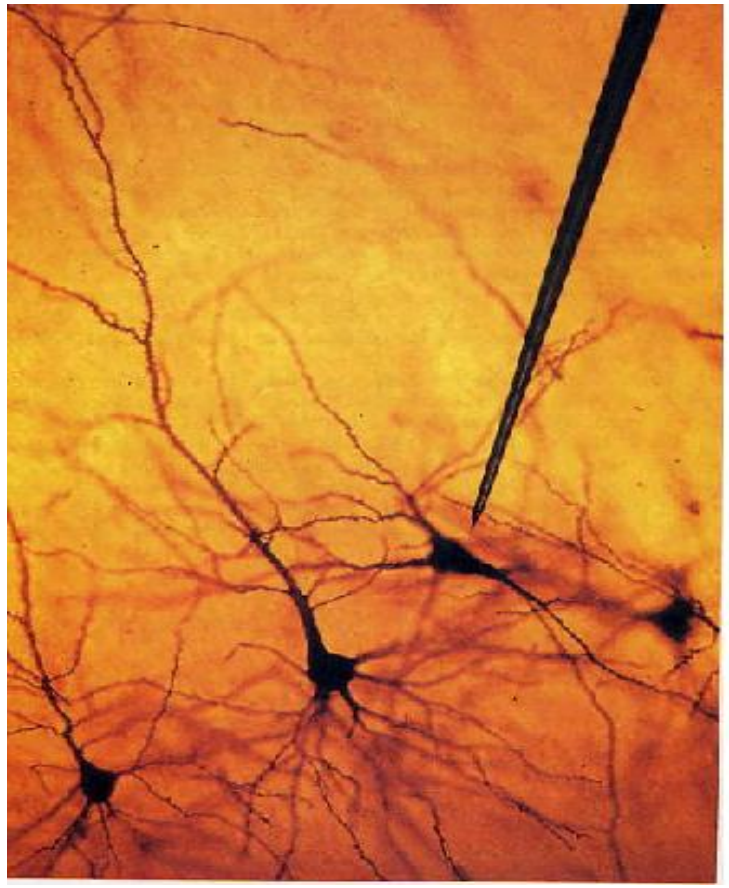
(D) Line drawing by 7-year old I. Lleras (2010 A.D.).

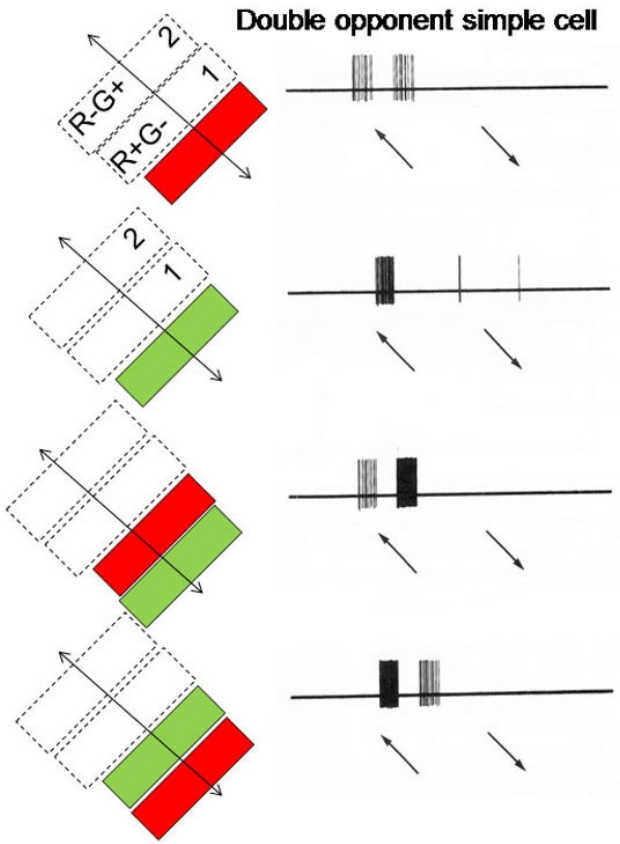





Hubel & Wiesel, 1960s

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

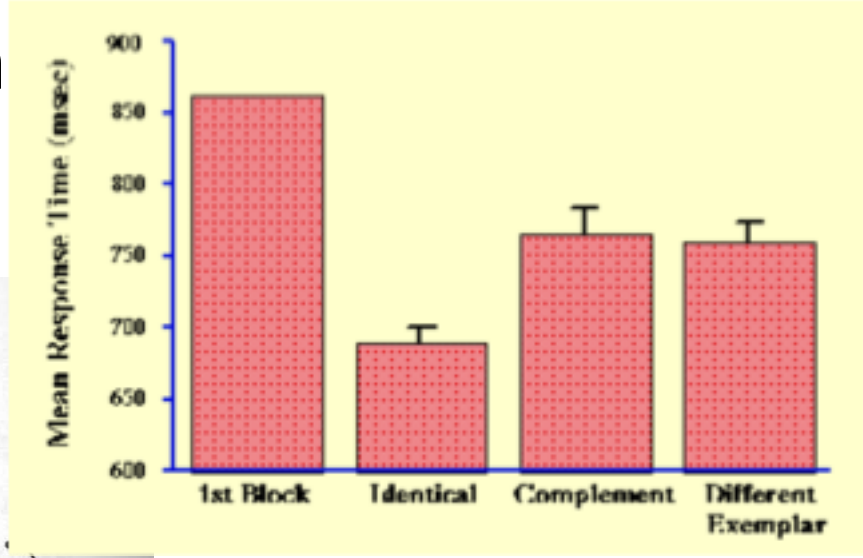





Hubel & Wiesel, 1960s

# We know edges are special from hum

(mammalian) vision studies




152 Biederman

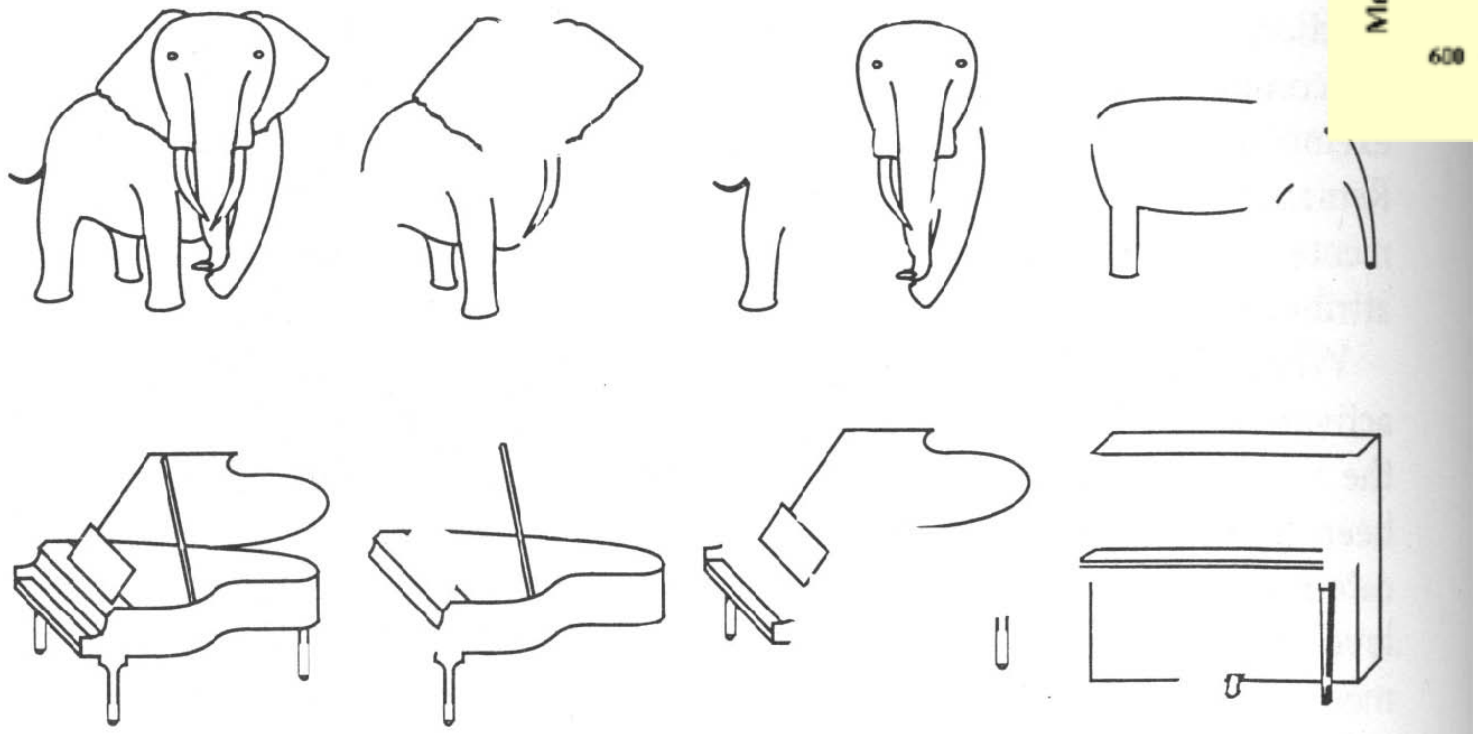
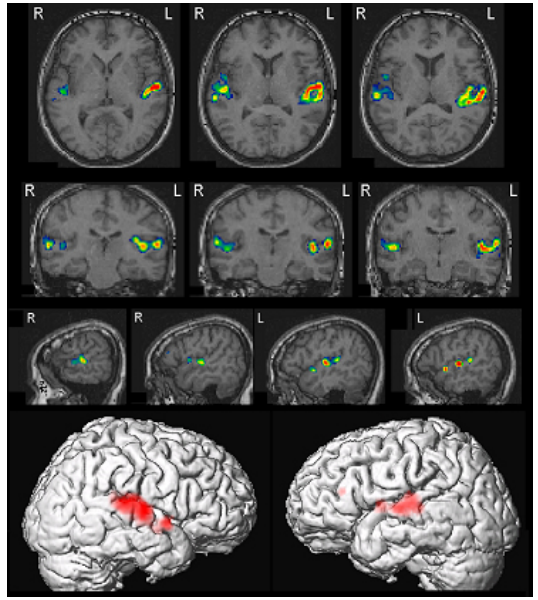
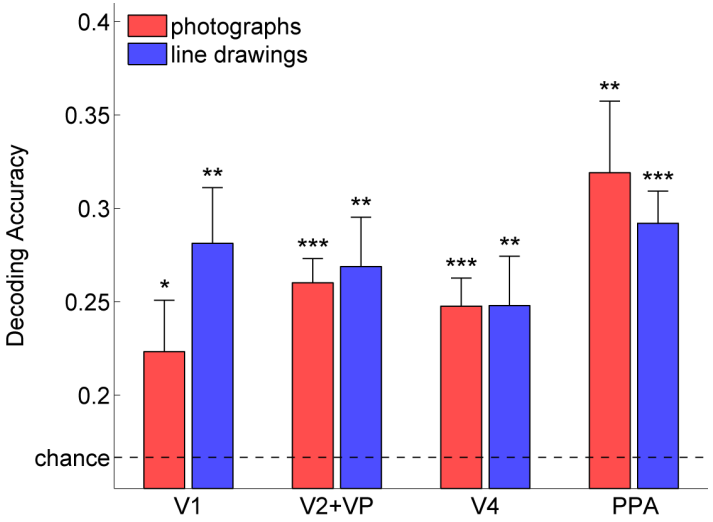


**Figure 4.14**
Complementary-part images. From an original intact image (left column), two complemen-




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

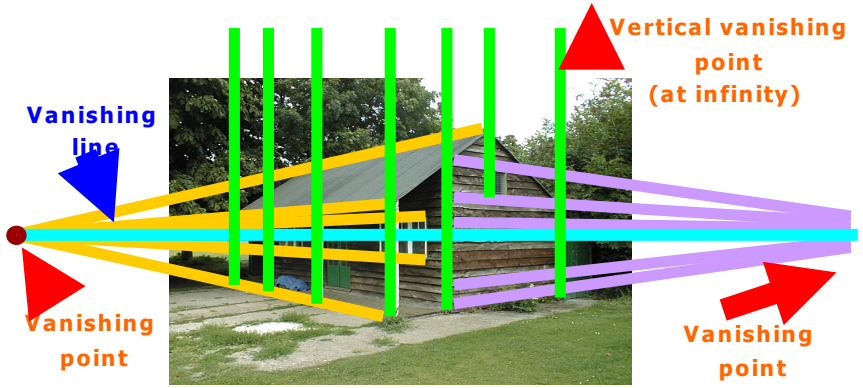
- **Ideal:** artist's line drawing (but artist is also using object-level knowledge)

Source: D. Lowe

# Why do we care about edges?

- Extract information, recognize objects

- Recover geometry and viewpoint




Source: J. Hayes

# Origins of edges

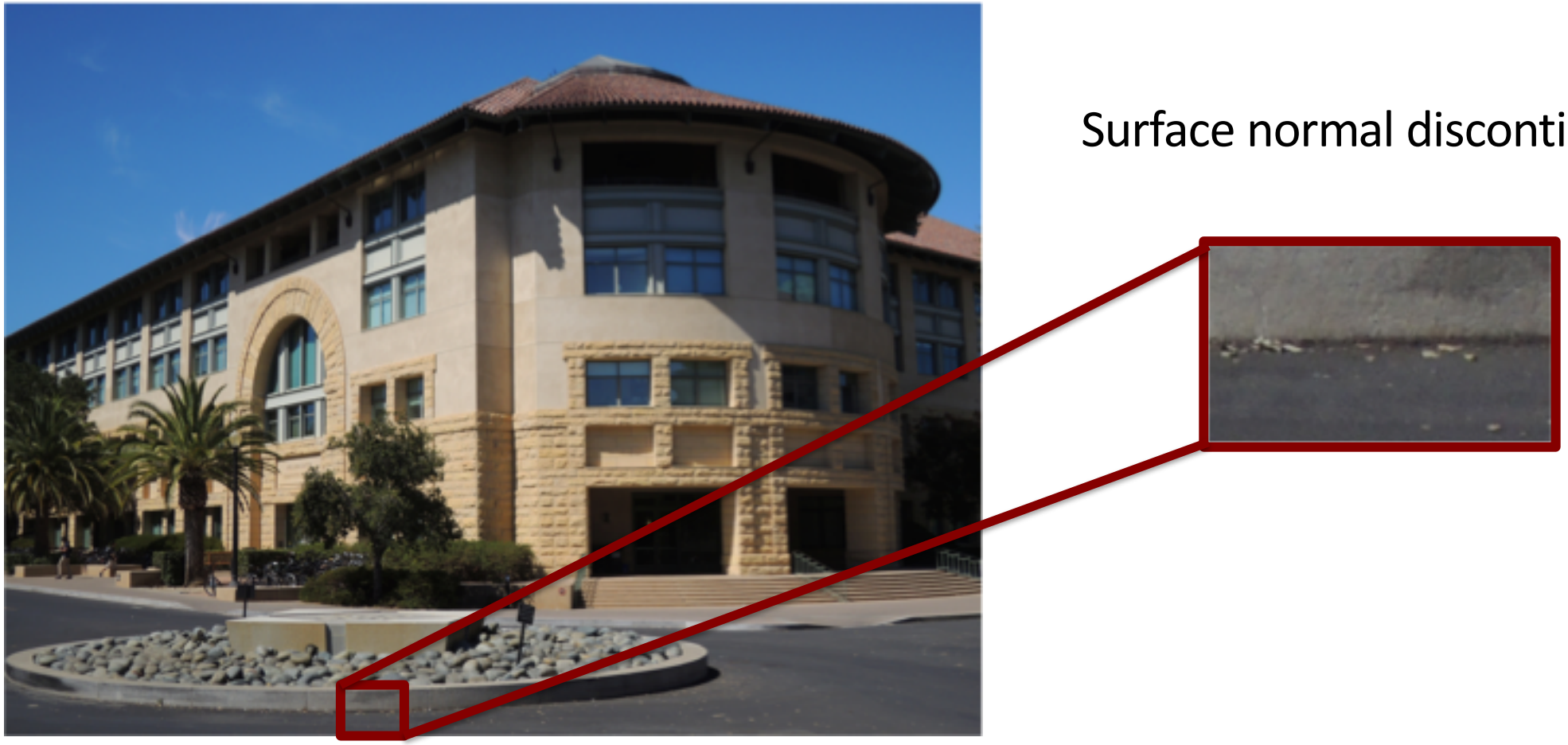
surface normal discontinuity

depth discontinuity

surface color discontinuity

illumination discontinuity

# Closeup of edges



Surface normal discontinuity

# Closeup of edges

# Closeup of edges

Surface color discontinuity

# What we will learn today

- Edge detection
- Image Gradients
- A simple edge detector
- Sobel edge detector
- Canny edge detector
- Hough Transform

# Derivatives in 1D

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

# Derivatives in 1D - example

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

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

# Derivatives in 1D - example

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

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

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

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

# Discrete Derivative in 1D

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

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

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

# Types of Discrete derivative in 1D

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

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

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

# 1D discrete derivate filters

- 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]

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

# 1D discrete derivate filters

- Backward filter: [0 1 -1]

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

# 1D discrete derivate filters

- Backward filter: [0 1 -1]

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

- Forward: [-1 1 0]

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

# 1D discrete derivate example

$$f(x) = 10 \quad 15 \quad 10 \quad 10 \quad 25 \quad 20 \quad 20 \quad 20$$

$$f'(x) = 0 \quad 5 \quad -5 \quad 0 \quad 15 \quad -5 \quad 0 \quad 0$$

# Discrete derivate in 2D

Given function $f(x, y)$

# Discrete derivate in 2D

Given function $f(x, y)$

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

# Discrete derivate in 2D

Given function $f(x, y)$

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

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

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

# 2D discrete derivative filters

What does this filter do?

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

# 2D discrete derivative filters

What about this filter?

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

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

# 2D discrete derivative - example

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

# 2D discrete derivative - example

What happens when we apply this filter?

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

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

# 2D discrete derivative - example

What happens when we apply this filter?

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

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

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

# 2D discrete derivative - example

Now let's try the other filter!

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

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

# 2D discrete derivative - example

What happens when we apply this filter?

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

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

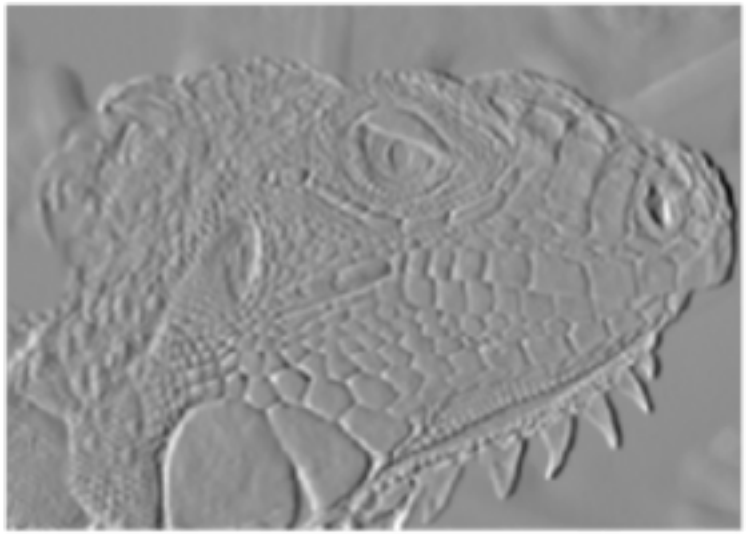
$$I_x = \begin{bmatrix} 0 & 10 & 10 & 0 & 0 \\ 0 & 10 & 10 & 0 & 0 \\ 0 & 10 & 10 & 0 & 0 \\ 0 & 10 & 10 & 0 & 0 \\ 0 & 10 & 10 & 0 & 0 \end{bmatrix}$$

# 3x3 image gradient filters

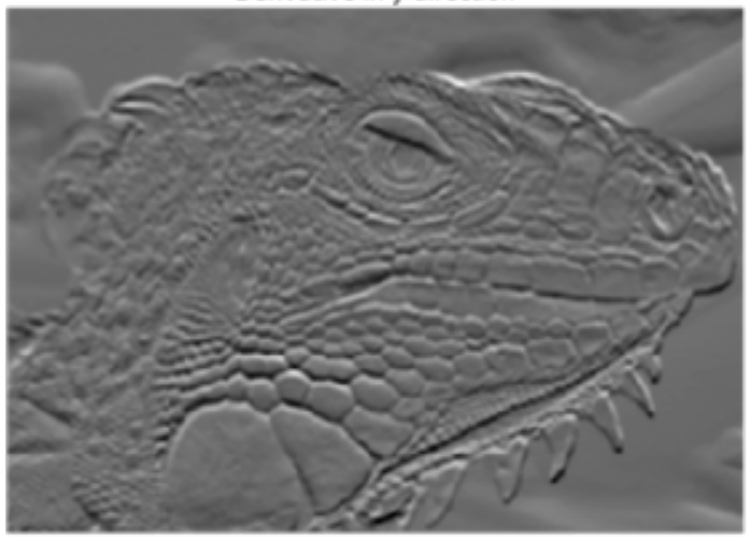
$$\frac{1}{3}\begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}$$

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

Derivative in x direction



Derivative in y direction

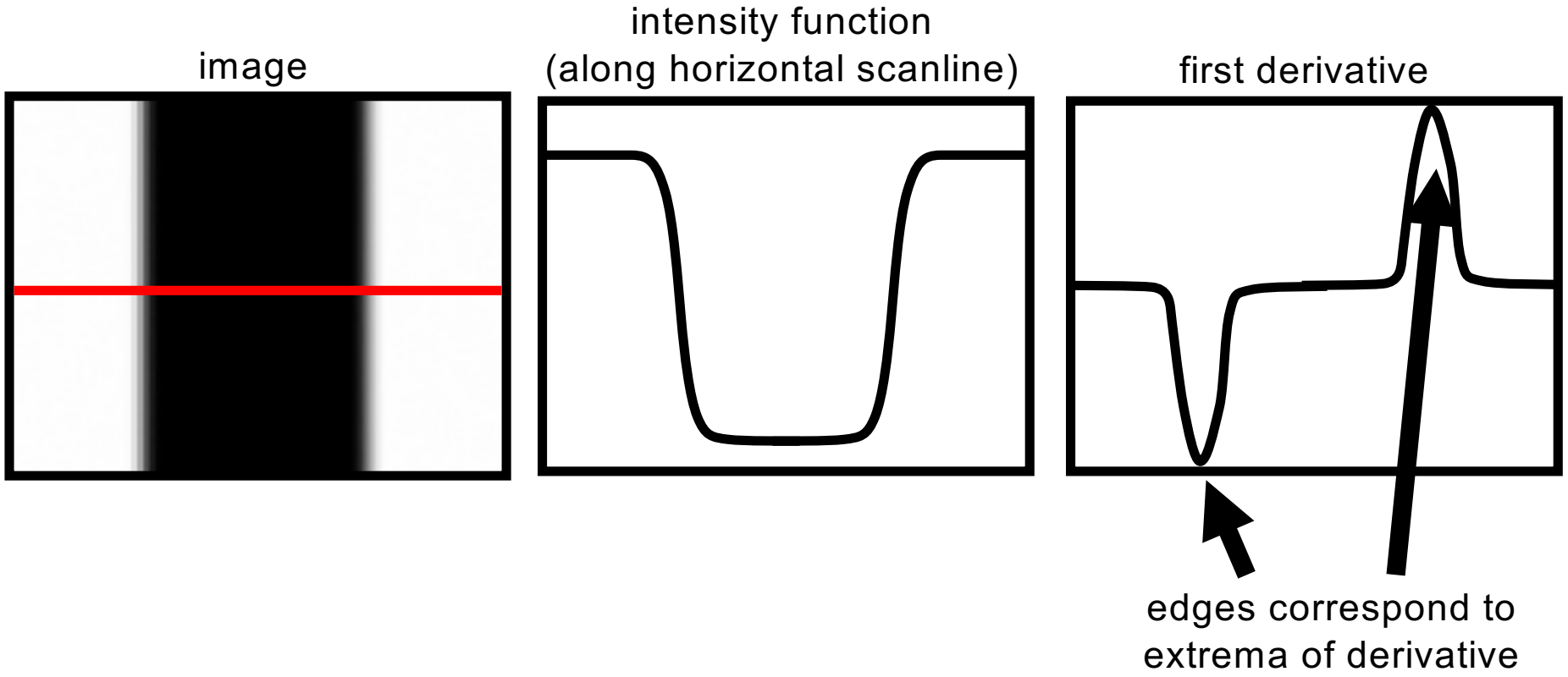


# What we will learn today

- Edge detection
- Image Gradients
- A simple edge detector
- Sobel edge detector
- Canny edge detector
- Hough Transform

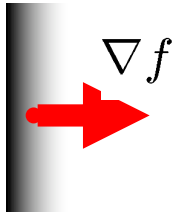
# Characterizing edges

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



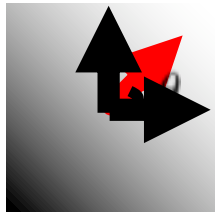
# Image gradient

- The gradient of an image: $\nabla f = \left[\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right]$



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

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



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

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

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

- how does this relate to the direction of the edge?

The *edge strength* is given by the gradient magnitude

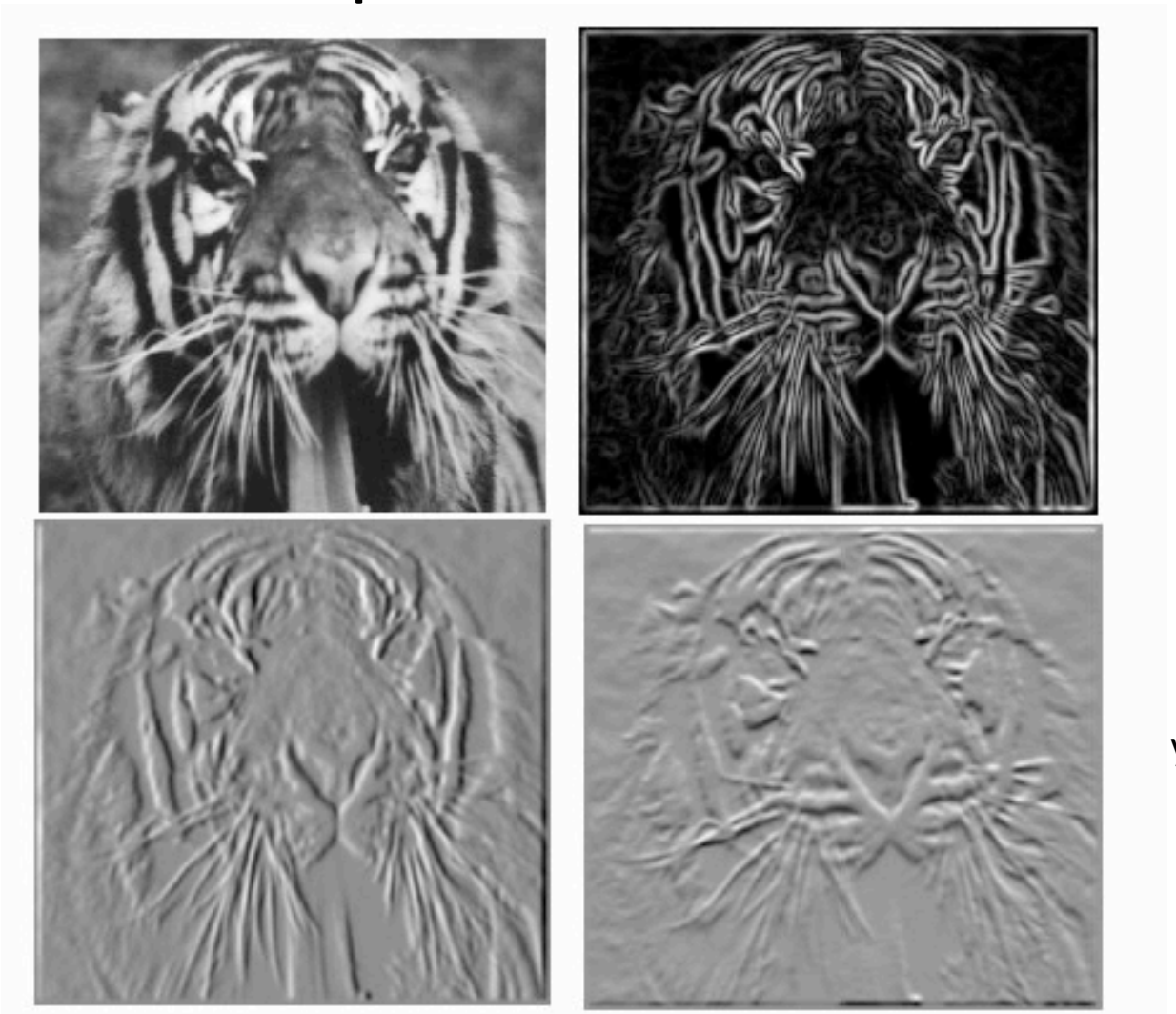
$$\|\nabla f\| = \sqrt{\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2}$$

Source: Steve Seitz

# Finite differences: example

Original Image

Gradient magnitude

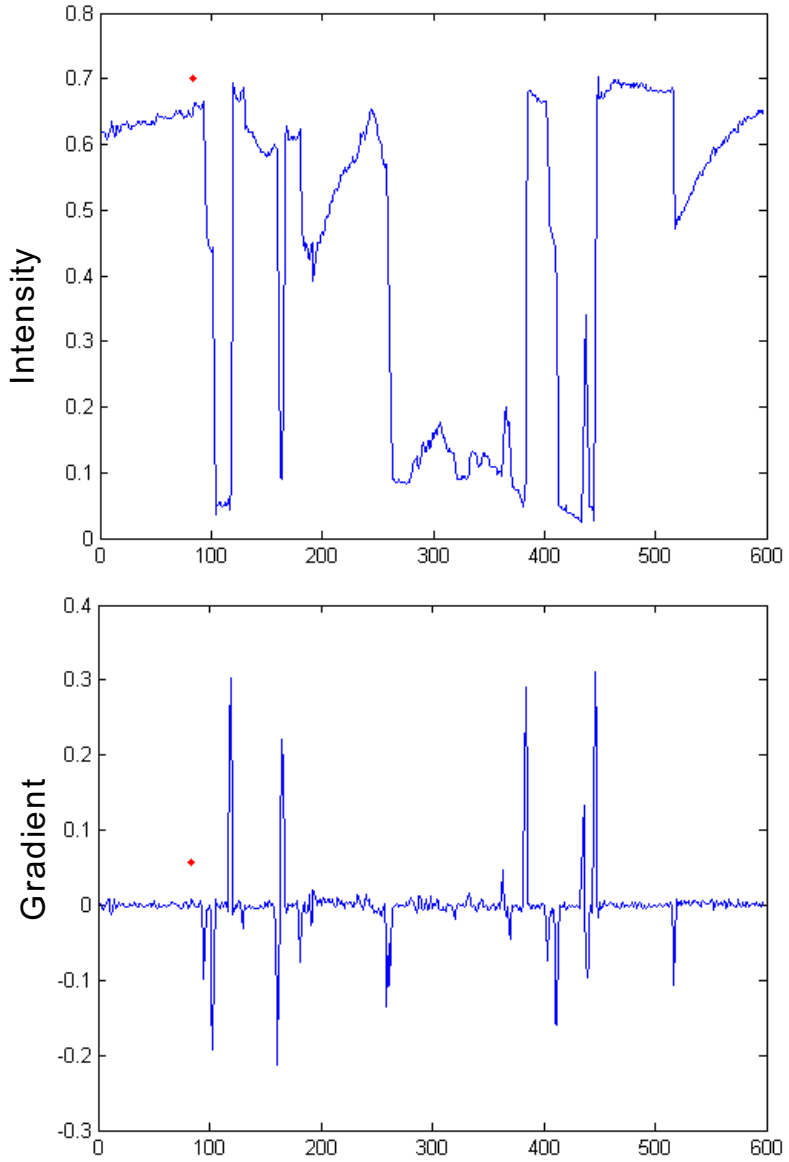


x-direction

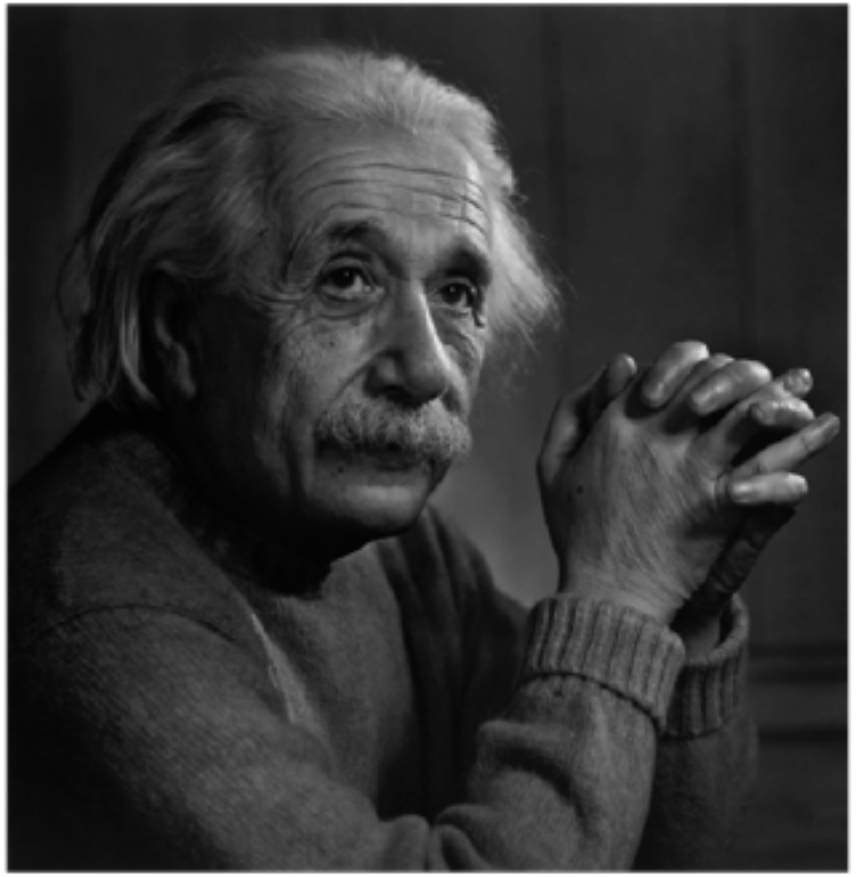
y-direction

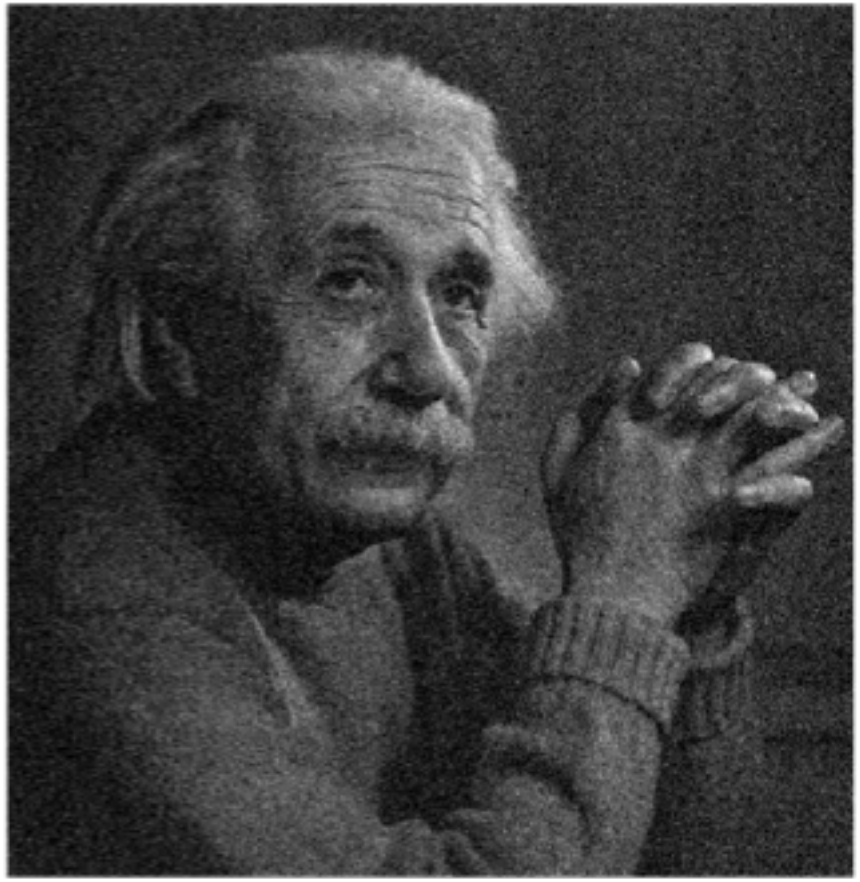
- Which one is the gradient in the x-direction? How about y-direction?

# Intensity profile



# Effects of noise

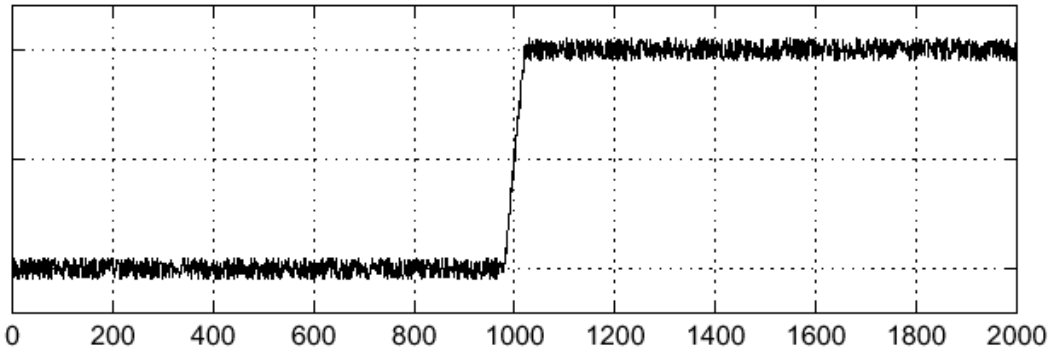




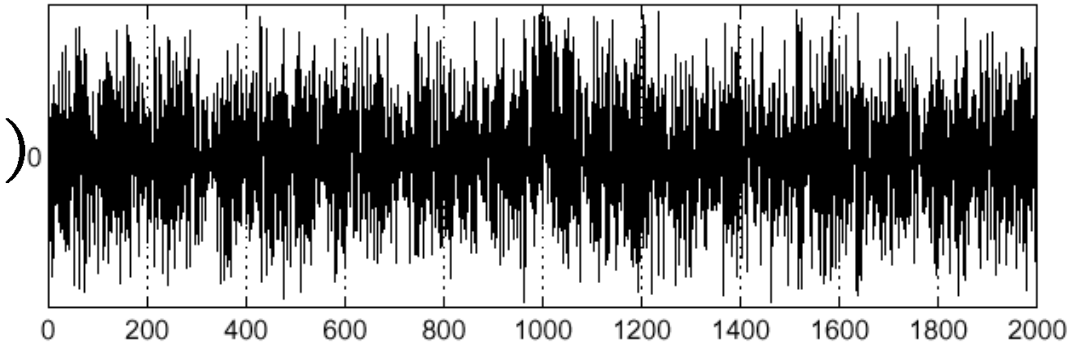
# Effects of noise

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

$f(x)$




$\frac{d}{dx}f(x)$




Where is the edge?

Source: S. Seitz

# 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?

Source: D. Forsyth

# 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

Source: D. Forsyth

# Smoothing with different filters

- Mean smoothing

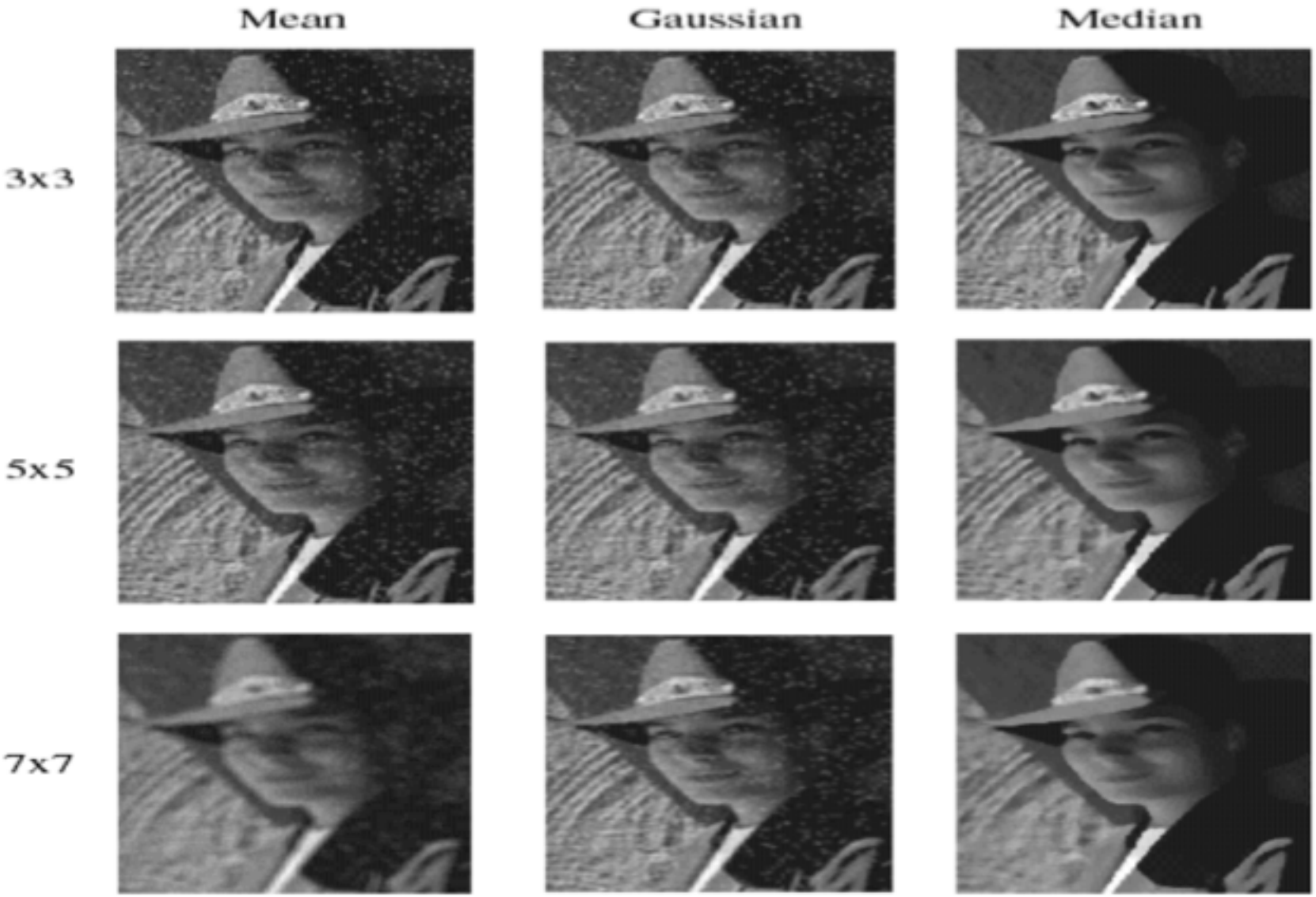
$$\begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \qquad \begin{bmatrix} 1 & 1 & 1 \end{bmatrix}$$

- Gaussian (smoothing * derivative)

$$\begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} \qquad \begin{bmatrix} 1 & 2 & 1 \end{bmatrix}$$

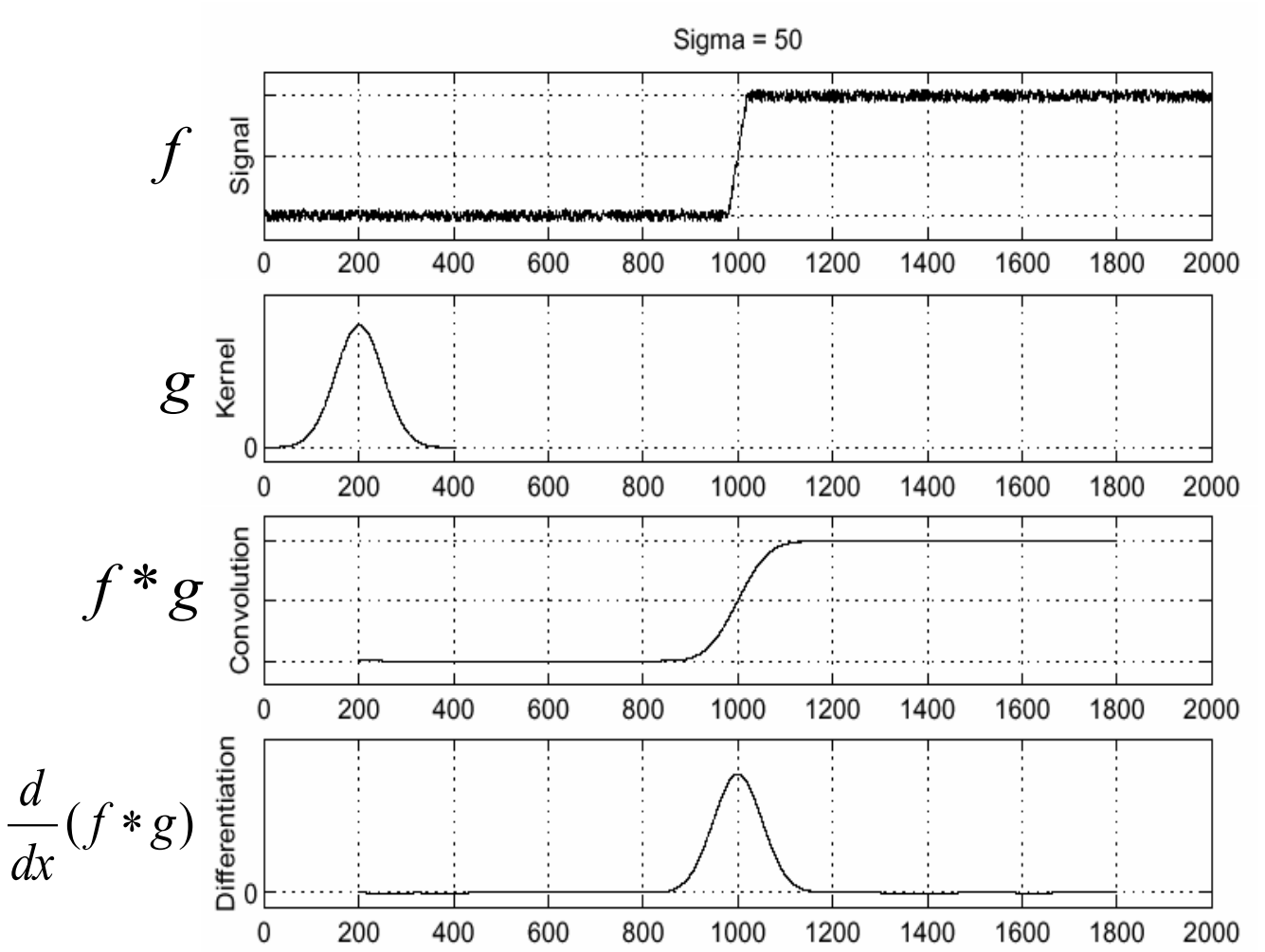
Slide credit: Steve Seitz

# Smoothing with different filters




Slide credit: Steve Seitz

# Solution: smooth first




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

Source: S. Seitz

# Derivative theorem of convolution

- This theorem gives us a very useful property:

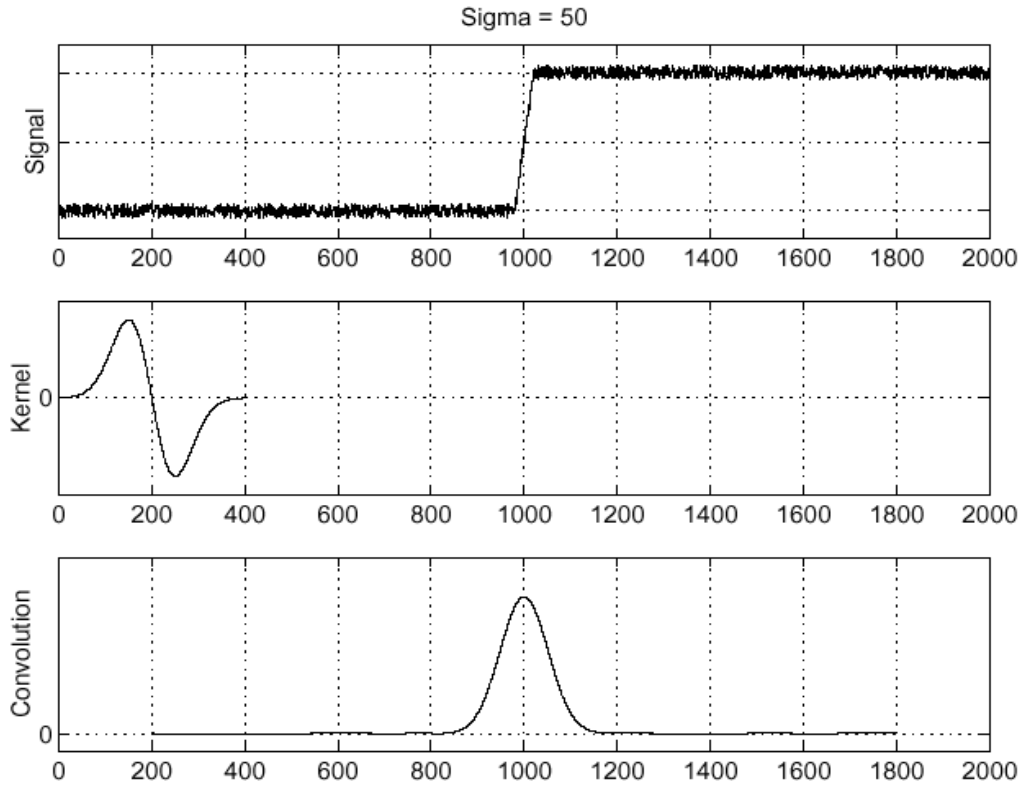
$$\frac{d}{dx}(f * g) = f * \frac{d}{dx} g$$

- This saves us one operation:

$f$

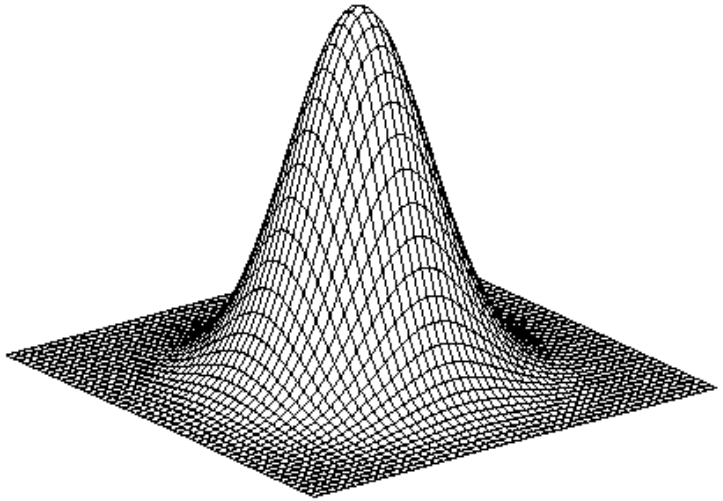
$\frac{d}{dx} g$

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

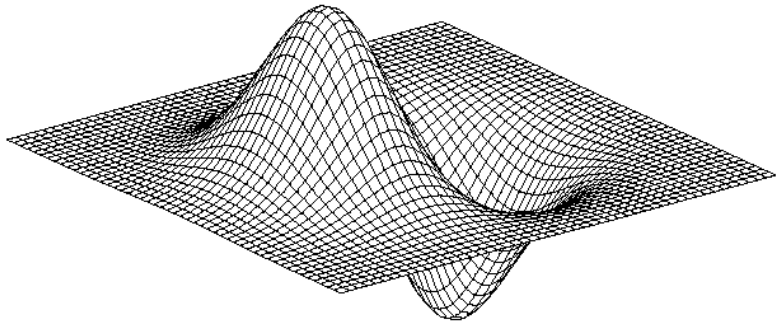



Source: S. Seitz

# Derivative of Gaussian filter



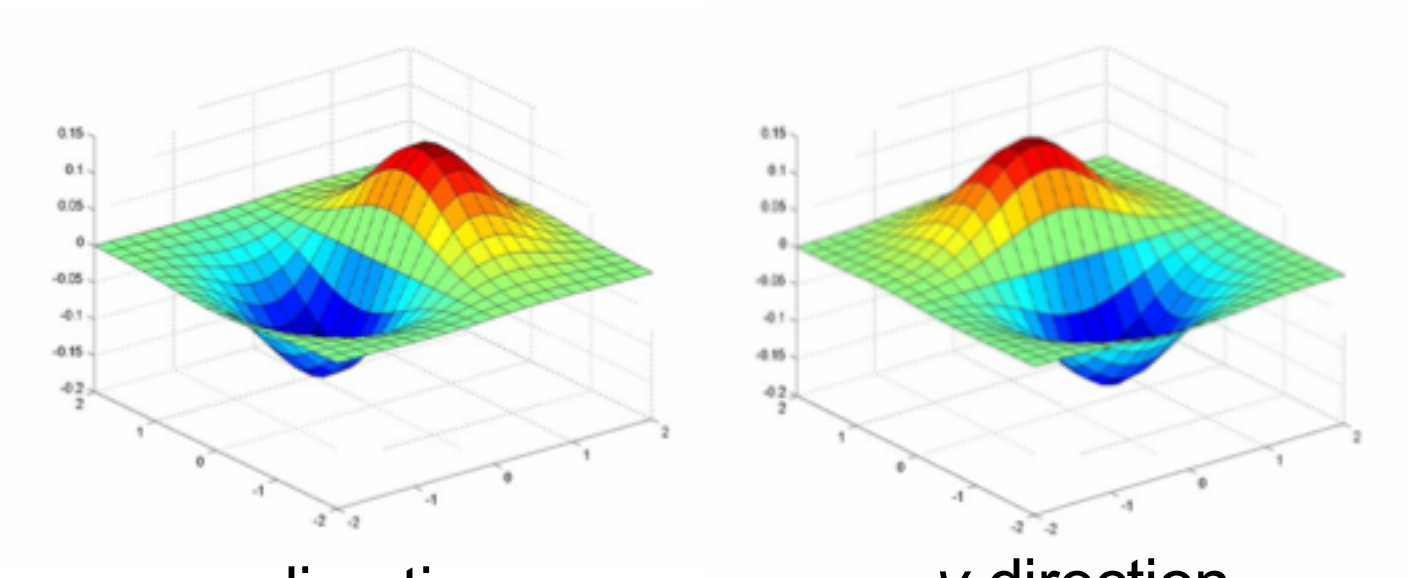
* [1 0 -1] =



2D-gaussian

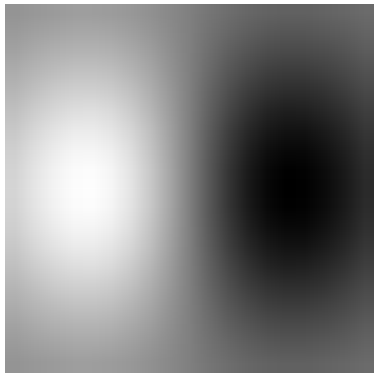
x - derivative

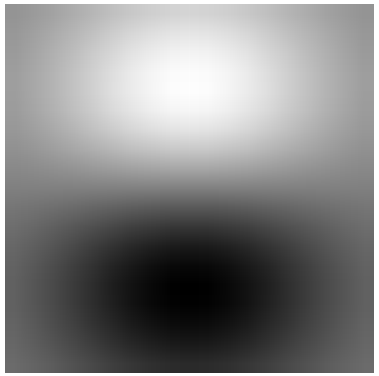
# Derivative of Gaussian filter



*x*-direction

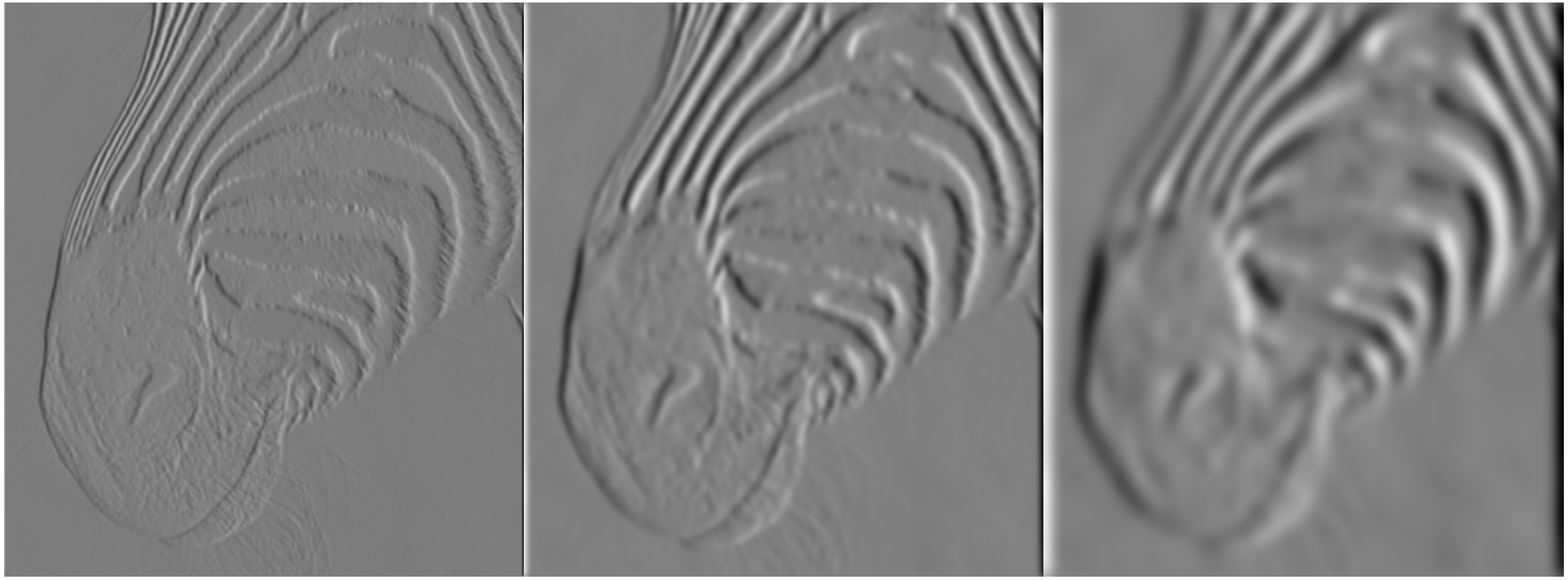
*y*-direction





# Derivative of Gaussian filter

# Tradeoff between smoothing at different scales



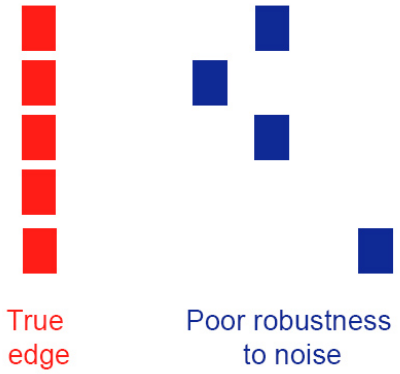
1 pixel 3 pixels 7 pixels

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

Source: D. Forsyth

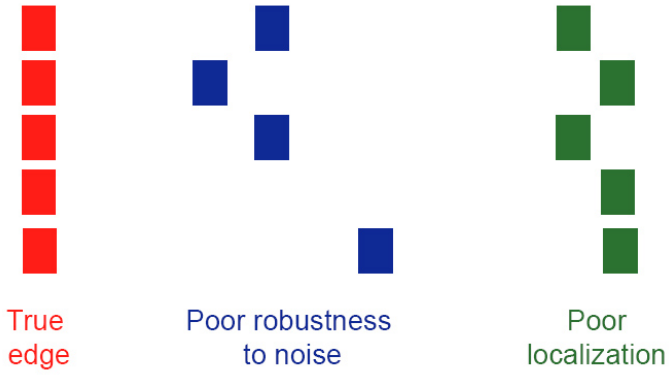
# Designing an edge detector

- 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)



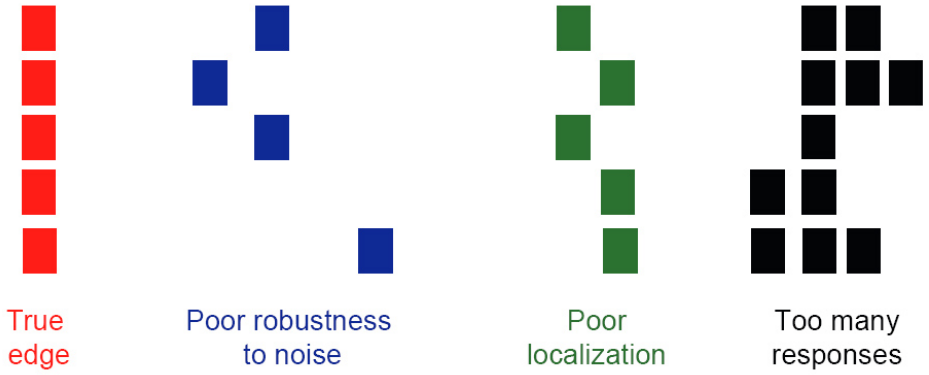
# Designing an edge detector

- 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
- Canny edge detector
- Hough transform

# 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

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

# Sobel Operation

- Smoothing + differentiation

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

Gaussian smoothing

differentiation

# Sobel Operation

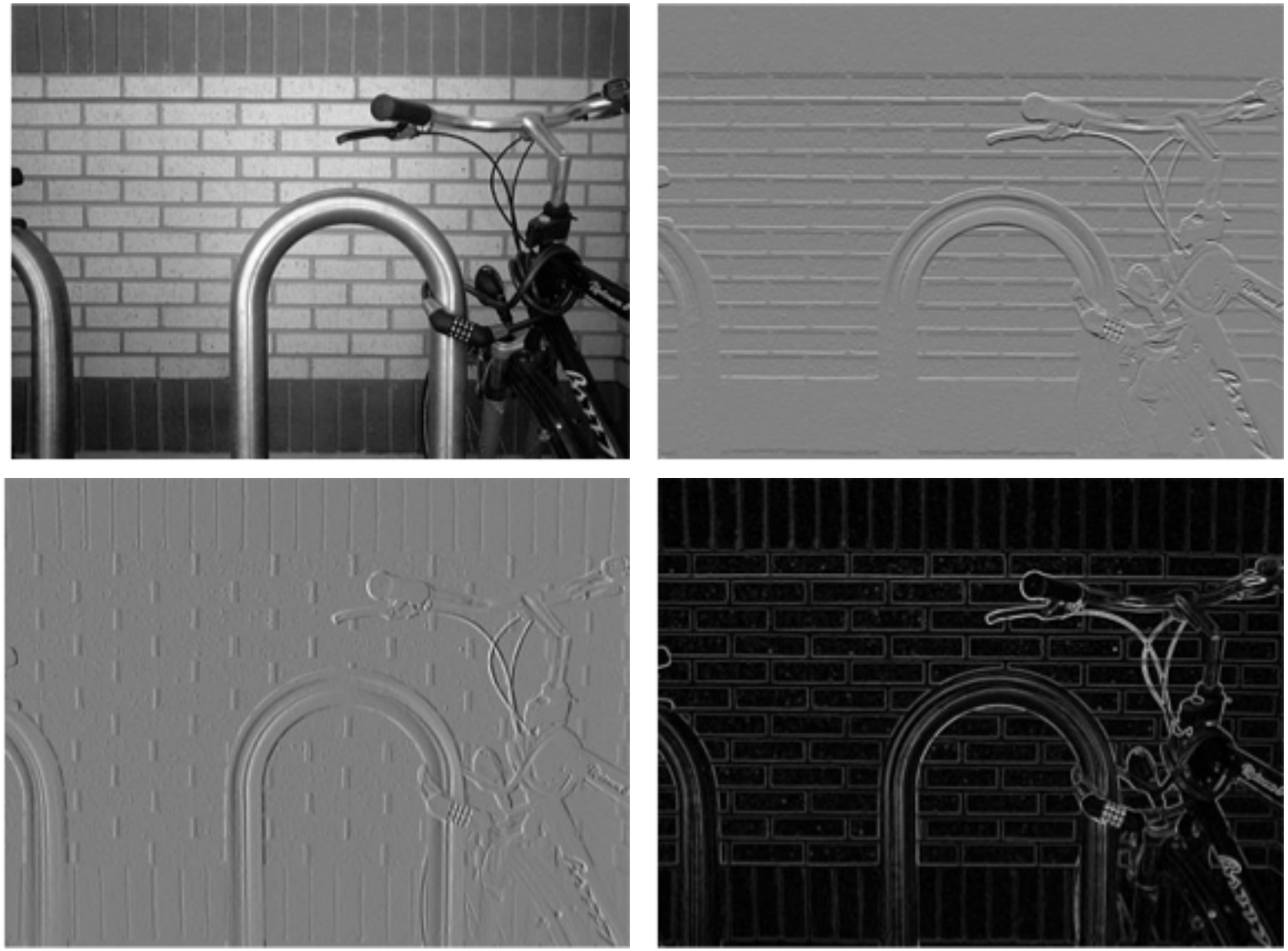
- Magnitude:

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

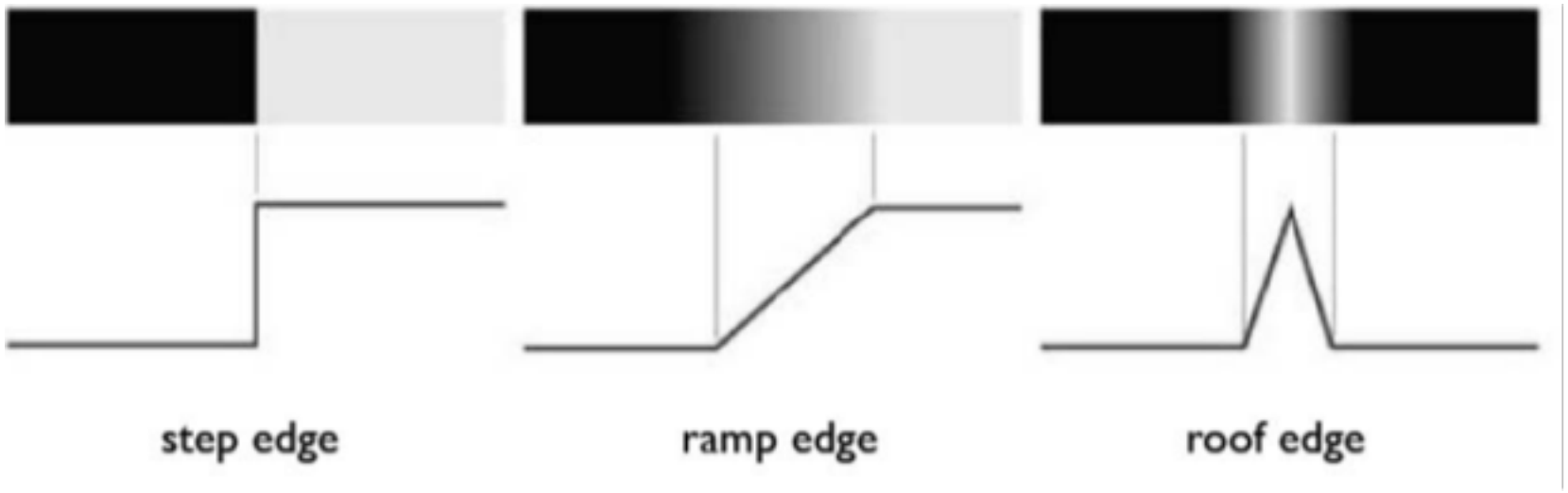
- Angle or direction of the gradient:

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

# Sobel Filter example



# 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
- Canny edge detector
- Hough Transform

# 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.

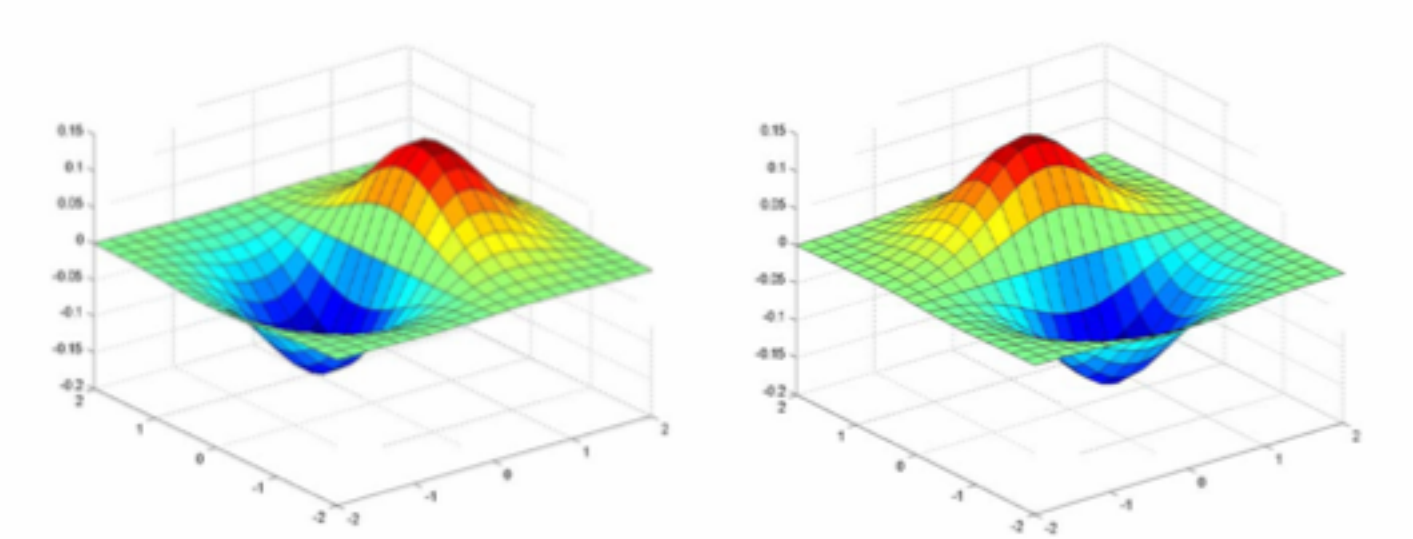
# Canny edge detector

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

# Example

- original image

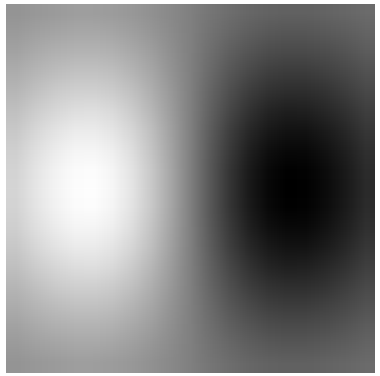
# Derivative of Gaussian filter

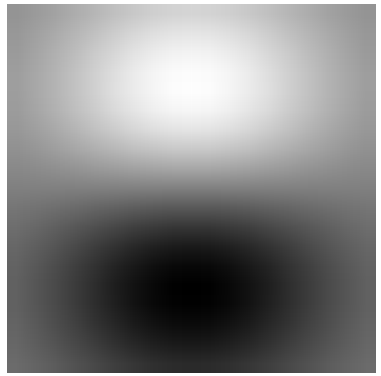


*x*-direction

*y*-direction

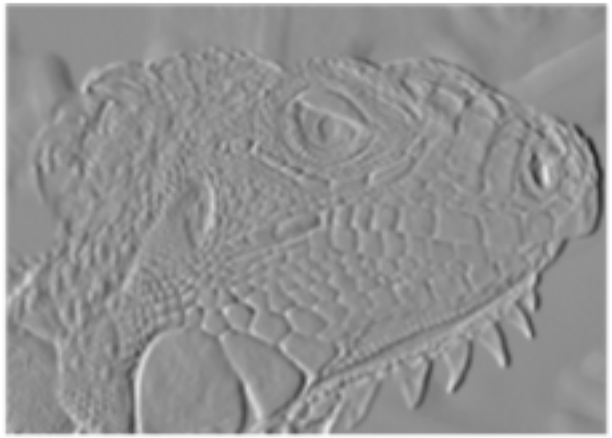
$$\begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix}$$



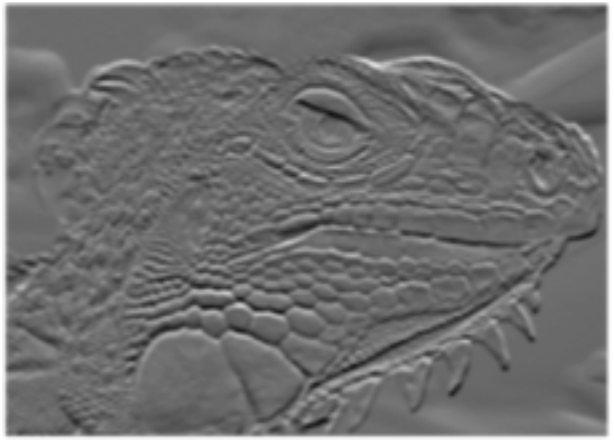


$$\begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

# Compute gradients (DoG)



X-Derivative of Gaussian

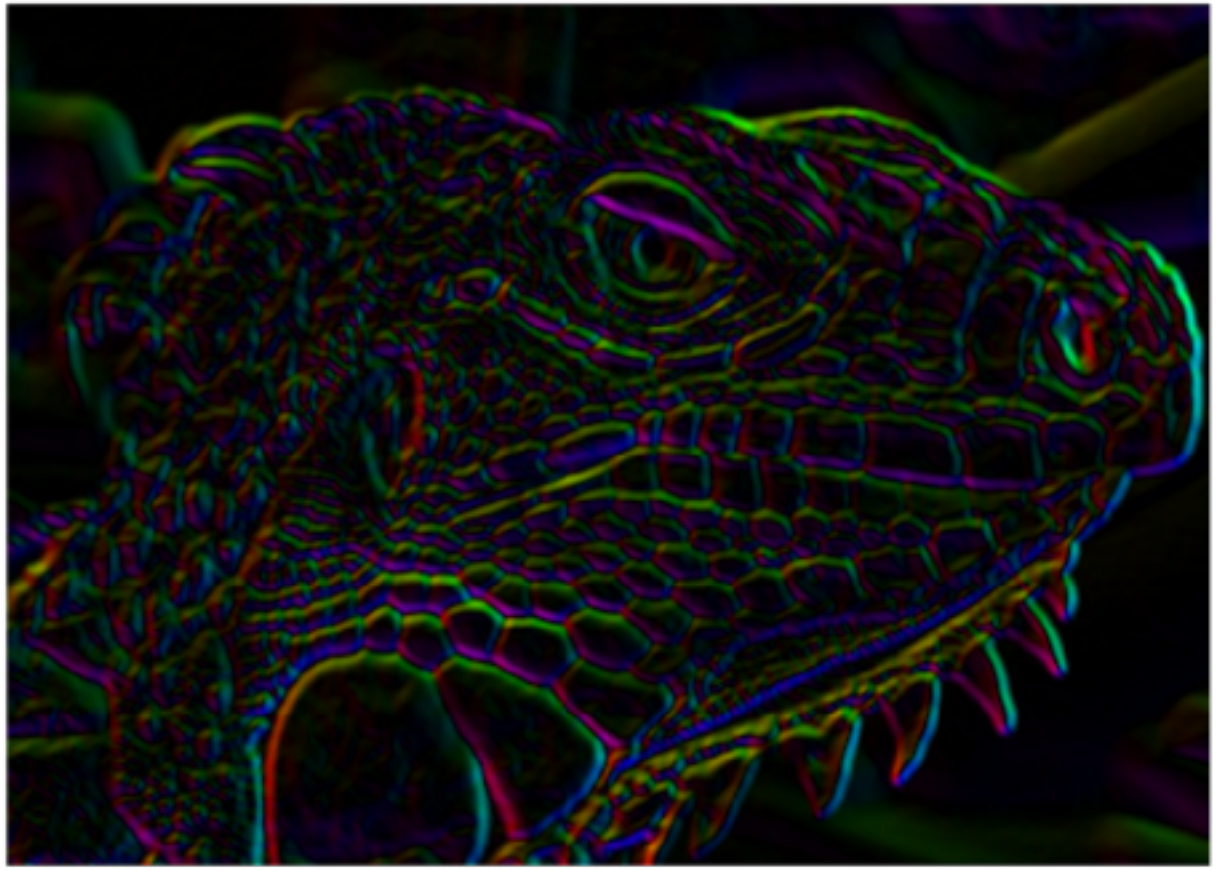


Y-Derivative of Gaussian

Gradient Magnitude

Source: J. Hayes

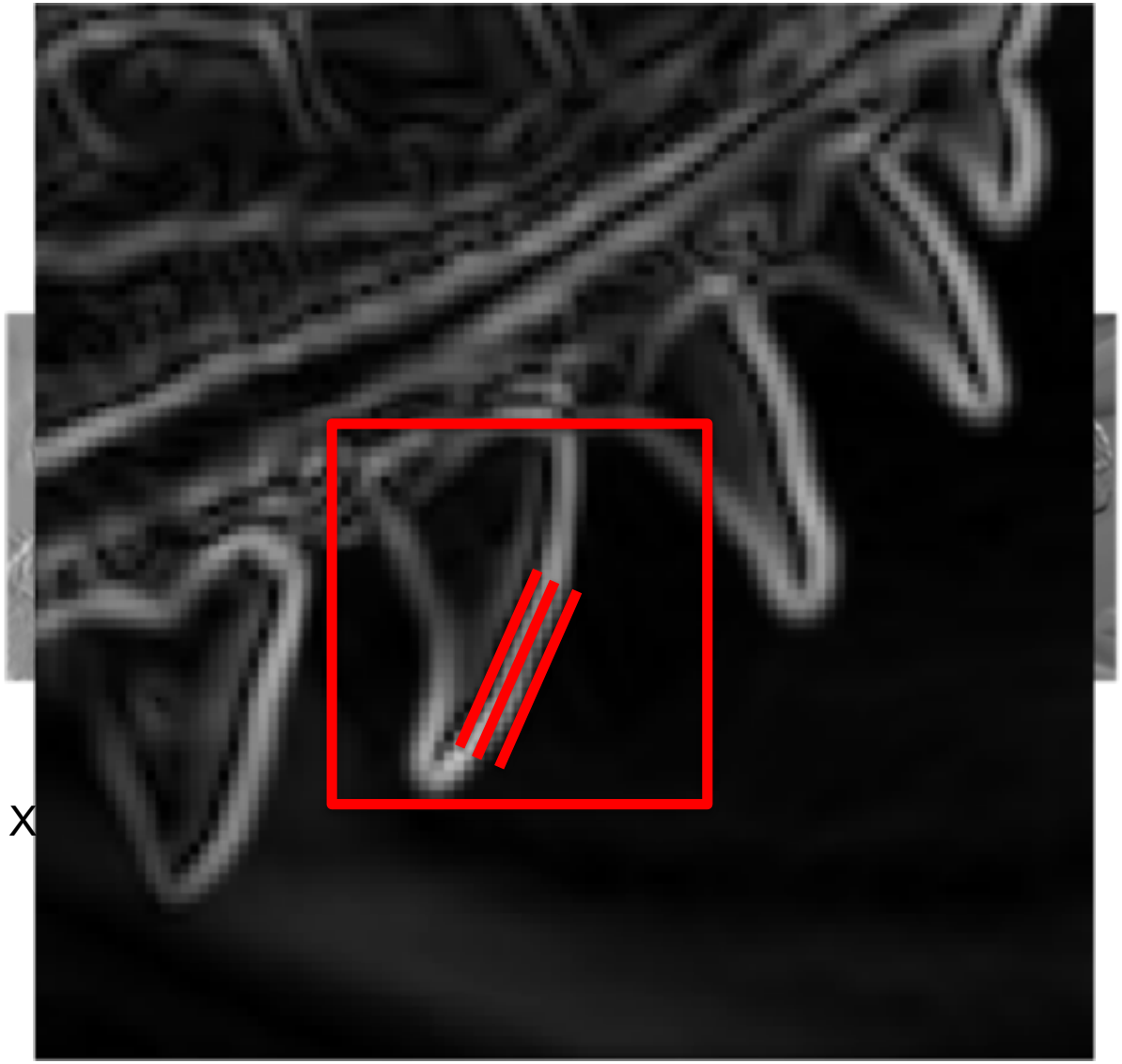
# Get orientation at each pixel



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

Source: J. Hayes

# Compute gradients (DoG)



X

Gradient Magnitude

# Canny edge detector

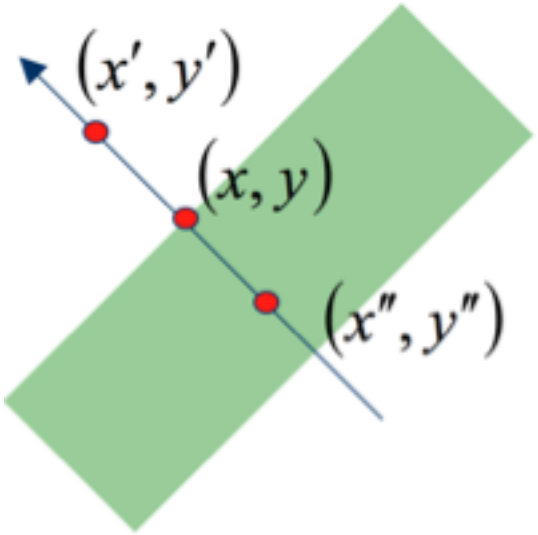
- Suppress Noise
- Compute gradient magnitude and direction
- Apply Non-Maximum Suppression
  - Assures minimal response

# Non-maximum suppression

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

# Remove spurious gradients

$|\nabla G|(x, y)$ is the gradient at pixel (x, y)




$$M(x,y) = \begin{cases} |\nabla G|(x,y) \text{ if } |\nabla G|(x,y) > |\nabla G|(x',y') \\ \qquad \& |\nabla G|(x,y) > |\nabla G|(x'',y'') \\ 0 \qquad \text{otherwise} \end{cases}$$

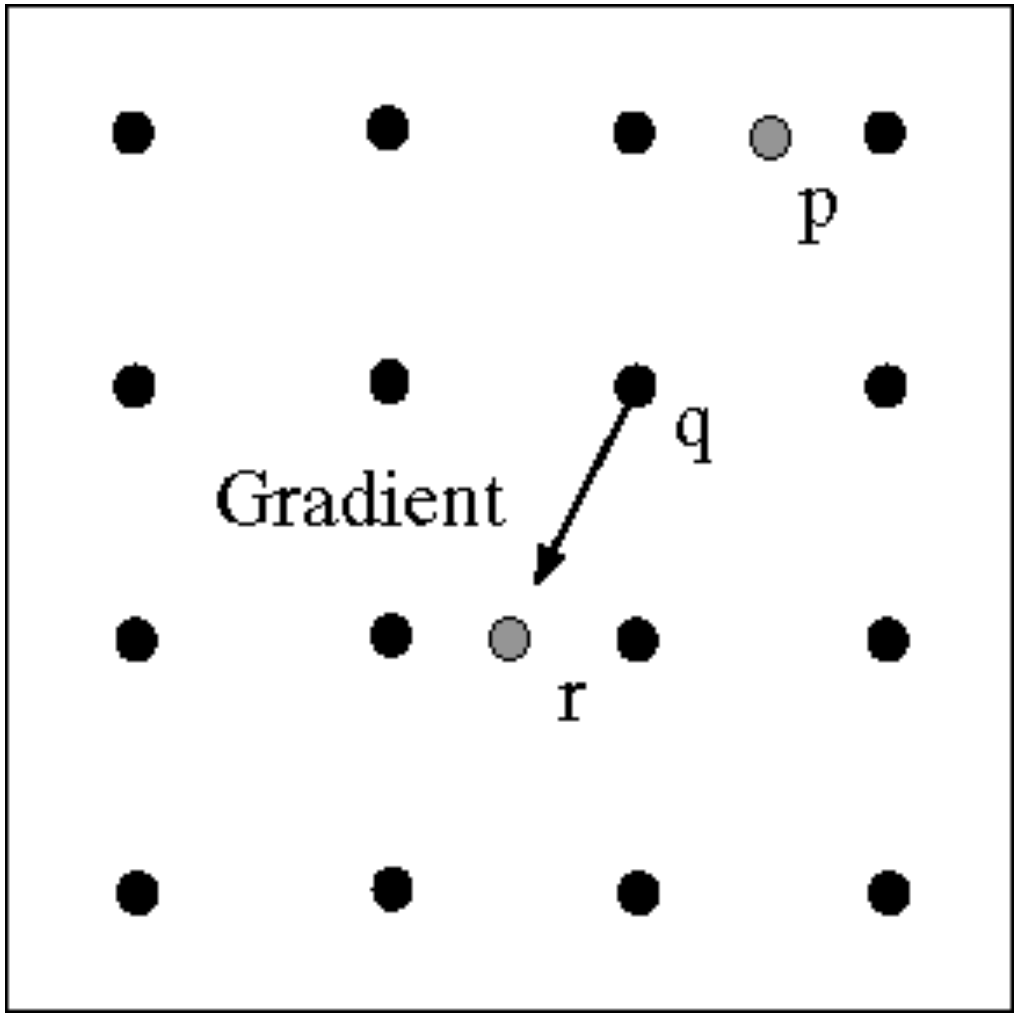
**x'** and **x"** are the neighbors of **x** along normal direction to an edge

Alper Yilmaz, Mubarak Shah Fall 2012, UCF

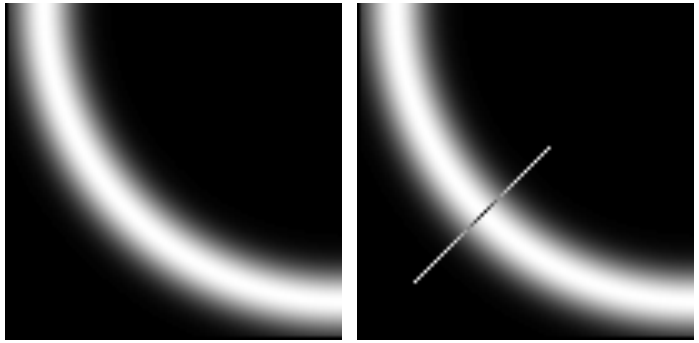
# Non-maximum suppression

- Edge occurs where gradient reaches a maxima
- Suppress non-maxima gradient even if it passes threshold
- 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

# Canny edge detector

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

# Hysteresis thresholding

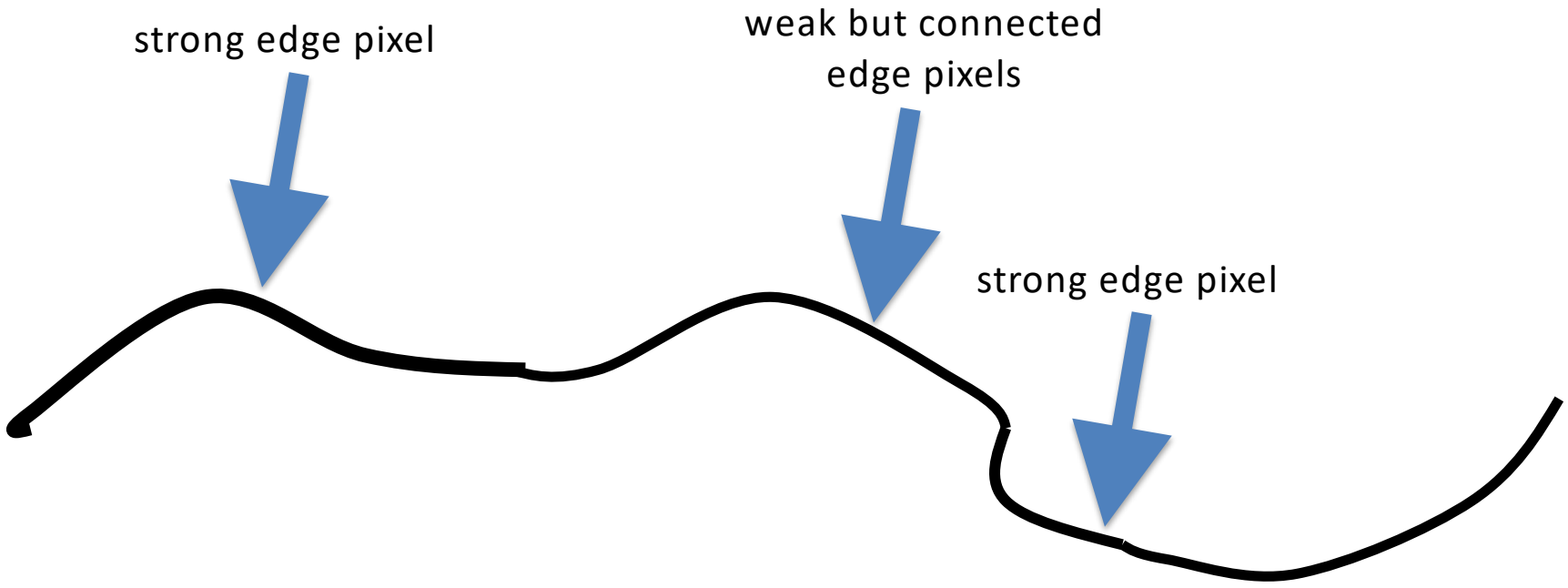
- Avoid streaking near threshold value
- Define two thresholds: Low and High
  - If less than Low, not an edge
  - If greater than High, strong edge
  - If between Low and High, weak edge

# Hysteresis thresholding

If the gradient at a pixel is

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

# Hysteresis thresholding




Source: S. Seitz

# Final Canny Edges

# Canny edge detector

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

# 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

Source: S. Seitz

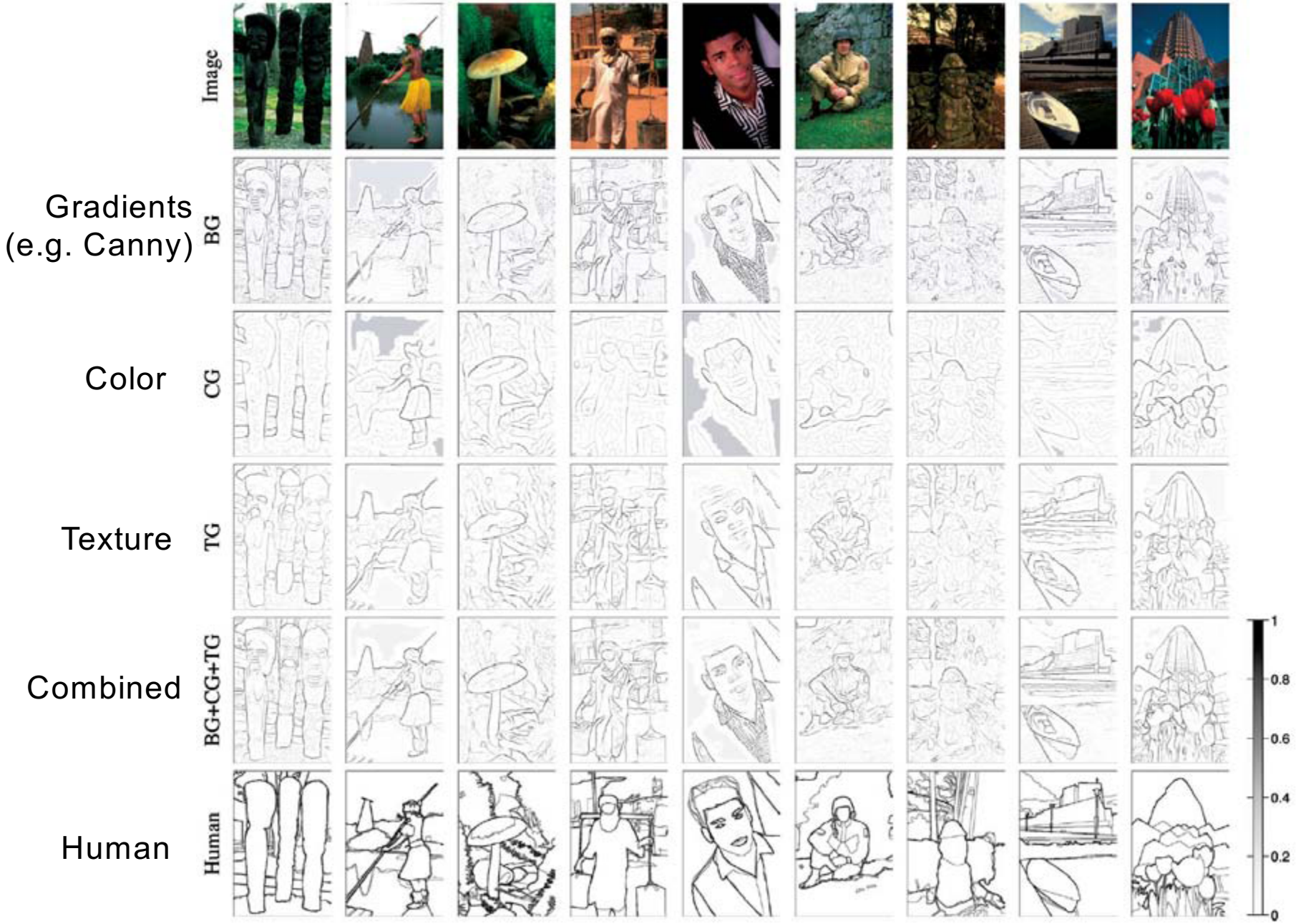
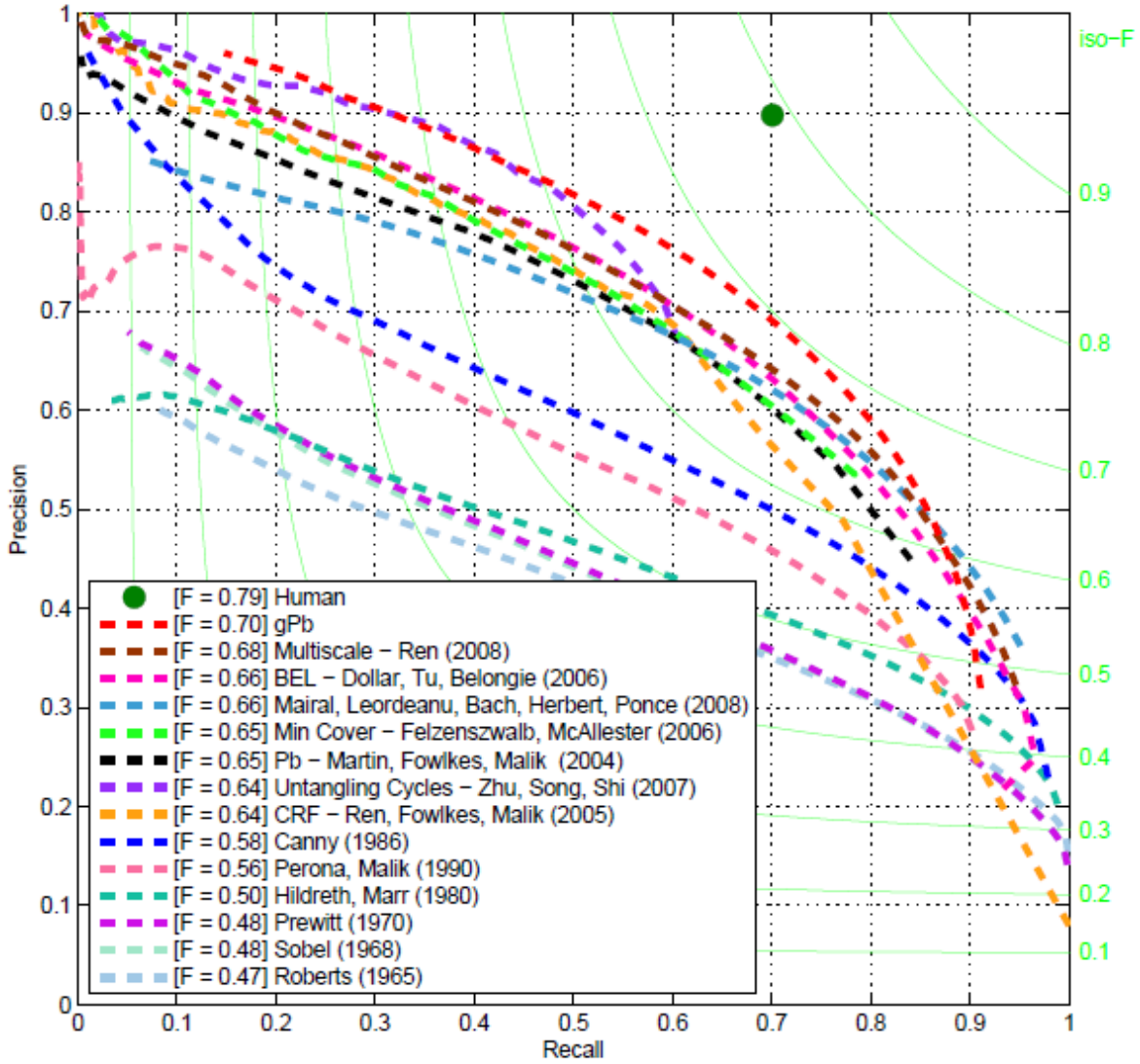


Image
Gradients
(e.g. Canny)
BG
Color
CG
Texture
TG
Combined
BG+CG+TG
Human
Human
1
0.8
0.6
0.4
0.2
0

# 45 years of boundary detection




Source: Arbelaez, Maire, Fowlkes, and Malik. TPAMI 2011 (pdf)

# What we will learn today

- Edge detection
- Image Gradients
- A simple edge detector
- Sobel Edge detector
- Canny edge detector
- Hough Transform

# Intro to Hough transform

- The Hough transform (HT) can be used to detect lines.
- It was introduced in 1962 (Hough 1962) and first used to find lines in images a decade later (Duda 1972).
- The goal is to find the location of lines in images.
- **Caveat**: Hough transform can detect lines, circles and other structures ONLY if their parametric equation is known.
- It can give robust detection under noise and partial occlusion

# Prior to Hough transform

- Assume that we have performed some edge detection, and a thresholding of the edge magnitude image.
- Thus, we have some pixels that may partially describe the boundary of some objects.

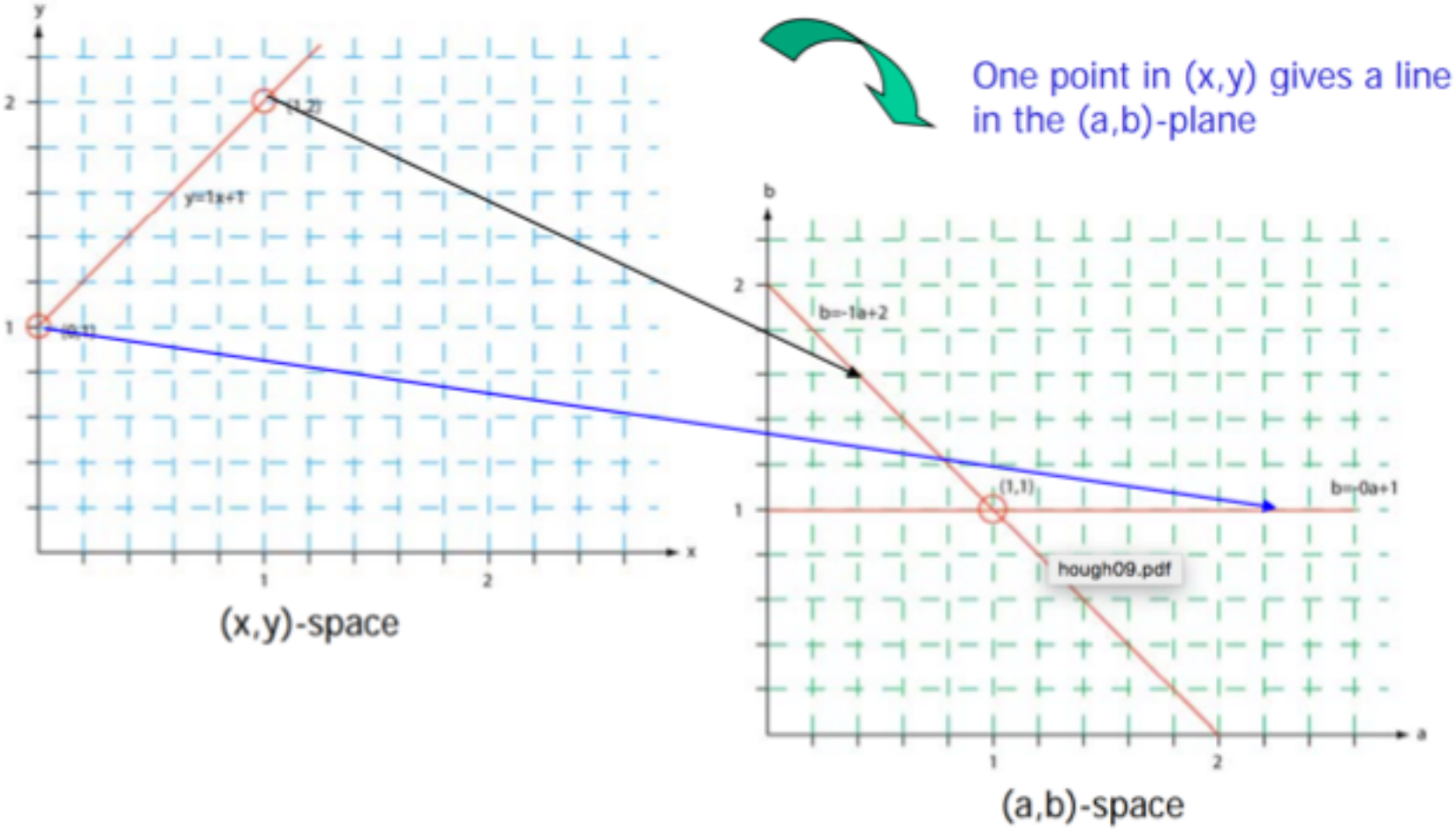
# Detecting lines using Hough transform

- We wish to find sets of pixels that make up straight lines.
- Consider a point of known coordinates $(x_i;y_i)$
  - There are many lines passing through the point $(x_i, y_i)$.
- Straight lines that pass that point have the form $y_i = a*x_i + b$
  - Common to them is that they satisfy the equation for some set of parameters (a, b)

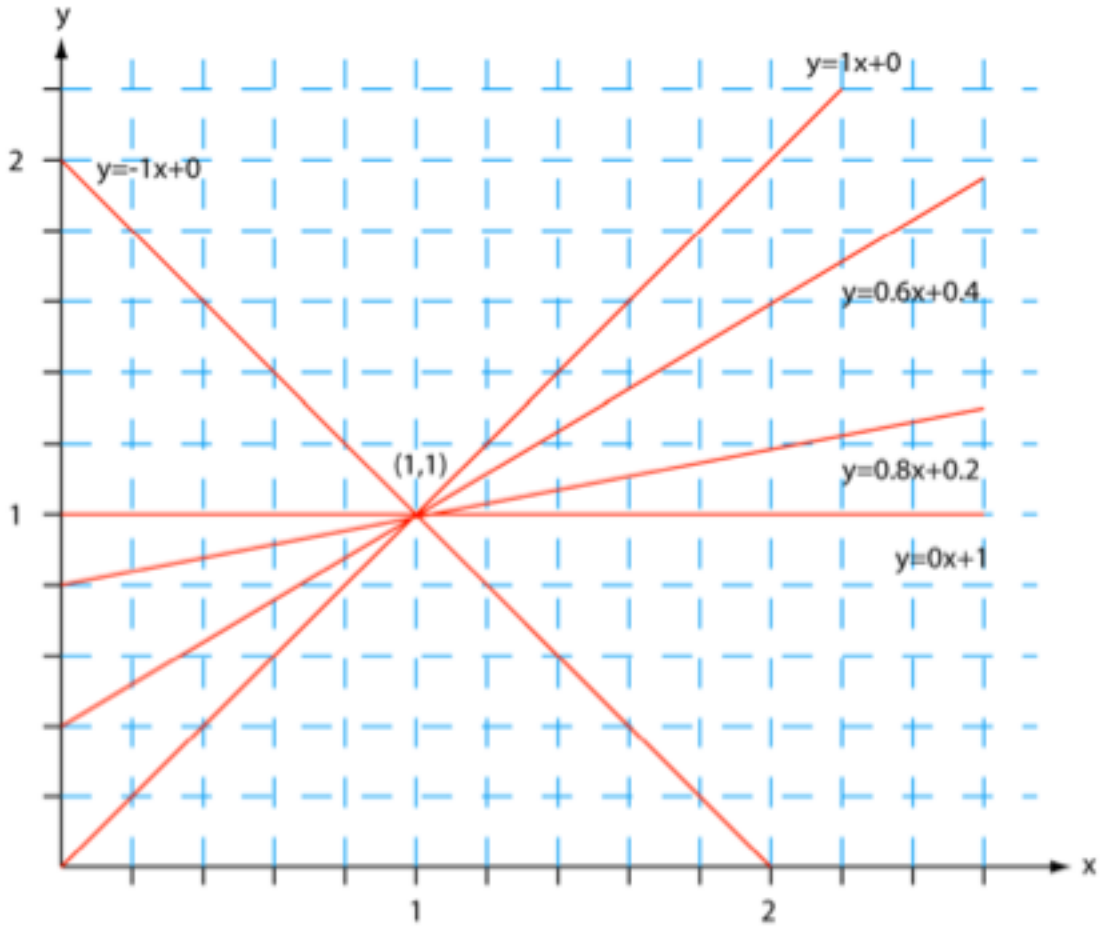
# Detecting lines using Hough transform

- This equation can obviously be rewritten as follows:
  - $b = -a*x_i + y_i$
  - We can now consider x and y as parameters
  - a and b as variables.
- This is a line in (a, b) space parameterized by x and y.
  - So: a single point in $x_1$,$y_1$-space gives a line in (a,b) space.
  - Another point ($x_2$, $y_2$ ) will give rise to another line (a,b) space.

# Detecting lines using Hough transform



# Detecting lines using Hough transform



# Detecting lines using Hough transform

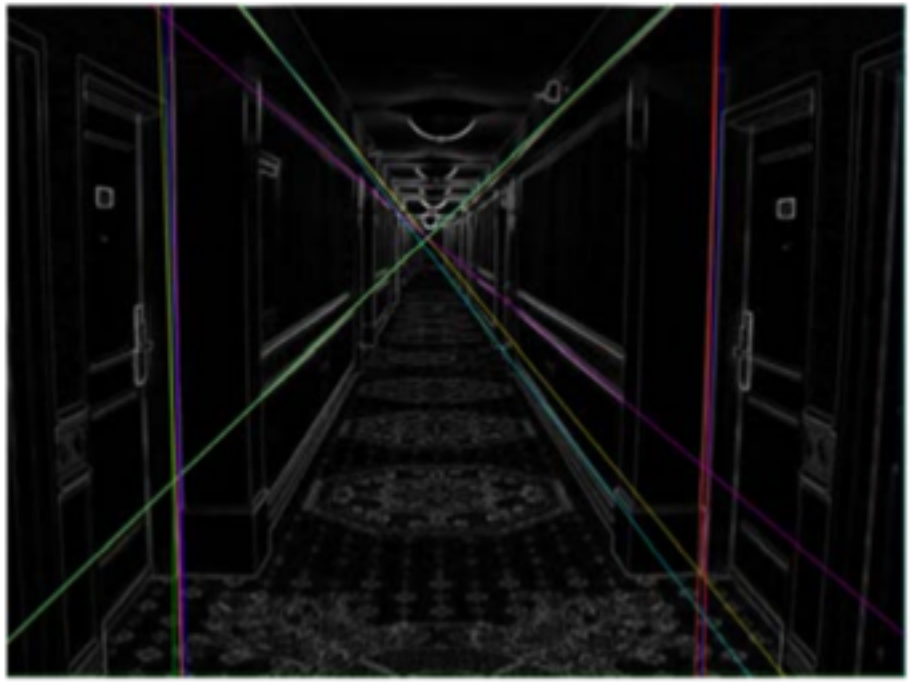
- Two points ($x_1$, $y_1$) and($x_2$ $y_2$) define a line in the (x, y) plane.
- These two points give rise to two different lines in (a,b) space.
- In (a,b) space these lines will intersect in a point (a' b')
- All points on the line defined by ($x_1$, $y_1$) and ($x_2$ , $y_2$) in (x, y) space will parameterize lines that intersect in (a', b') in (a,b) space.

# Algorithm for Hough transform

- Quantize the parameter space (a b) by dividing it into cells
- This quantized space is often referred to as the accumulator cells.
- Count the number of times a line intersects a given cell.
  - For each pair of points ($x_1$, $y_1$) and ($x_2$, $y_2$) detected as an edge, find the intersection (a',b') in (a, b)space.
  - Increase the value of a cell in the range [[$a_{min}$, $a_{max}$],[$b_{min}$,$b_{max}$]] that (a', b') belongs to.
  - Cells receiving more than a certain number of counts (also called 'votes') are assumed to correspond to lines in (x,y) space.

# Output of Hough transform

- Here are the top 20 most voted lines in the image:

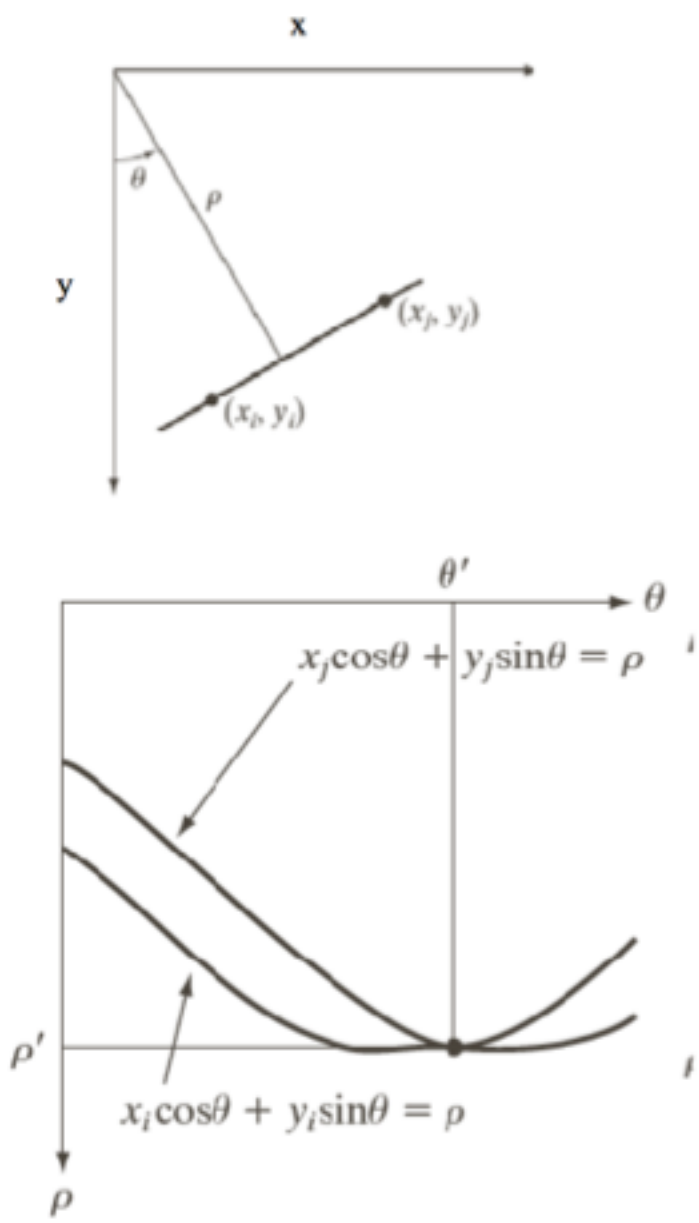


# Other Hough transformations

- We can represent lines as polar coordinates instead of $y = a*x + b$
- Polar coordinate representation:
  - $x*\cos\theta + y*\sin\theta = \rho$
- Can you figure out the relationship between
  - $(x\ y)$ and $(\rho\ \theta)$?

# Other Hough transformations

- Note that lines in (x y) space are not lines in (ρ θ) space, unlike (a b) space.
- A horizontal line will have θ=0 and ρ equal to the intercept with the y-axis.
- A vertial line will have θ=90 and ρ equal to the intercept with the x-axis.



# Example video

- https://youtu.be/4zHbI-fFIlI?t=3m35s

# Concluding remarks

- Advantages:
  - Conceptually simple.
  - Easy implementation
  - Handles missing and occluded data very gracefully.
  - Can be adapted to many types of forms, not just lines
- Disadvantages:
  - Computationally complex for objects with many parameters.
  - Looks for only one single type of object
  - Can be “fooled” by “apparent lines”.
  - The length and the position of a line segment cannot be determined.
  - Co-linear line segments cannot be separated.

# What we will learn today

- Edge detection
- Image Gradients
- A simple edge detector
- Sobel Edge detector
- Canny edge detector
- Hough Transform