Lecture 4:
Edge Detection

Harbin Institute of Technology (Shenzhen)

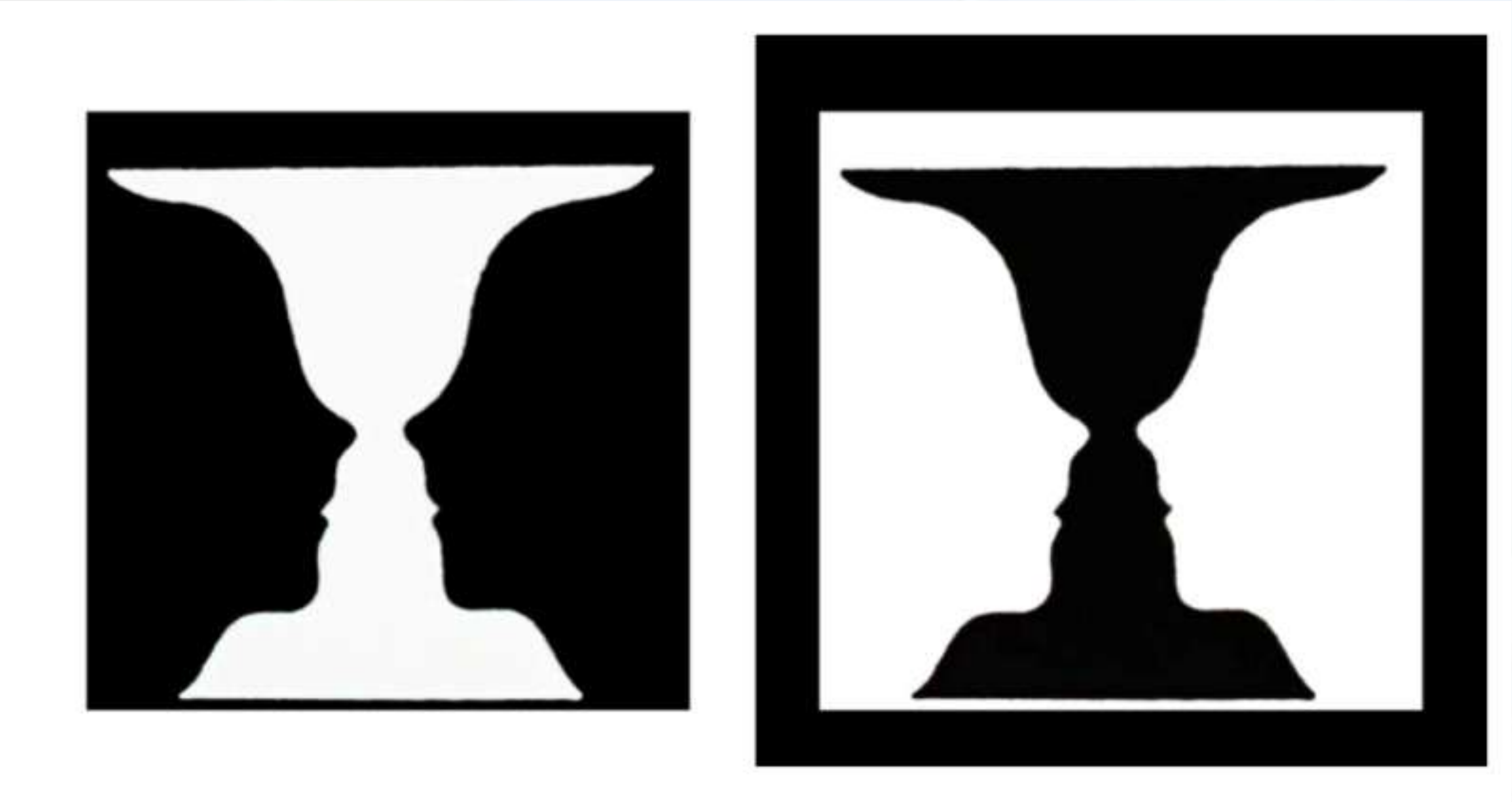
Jingyong Su

sujingyong@hit.edu.cn

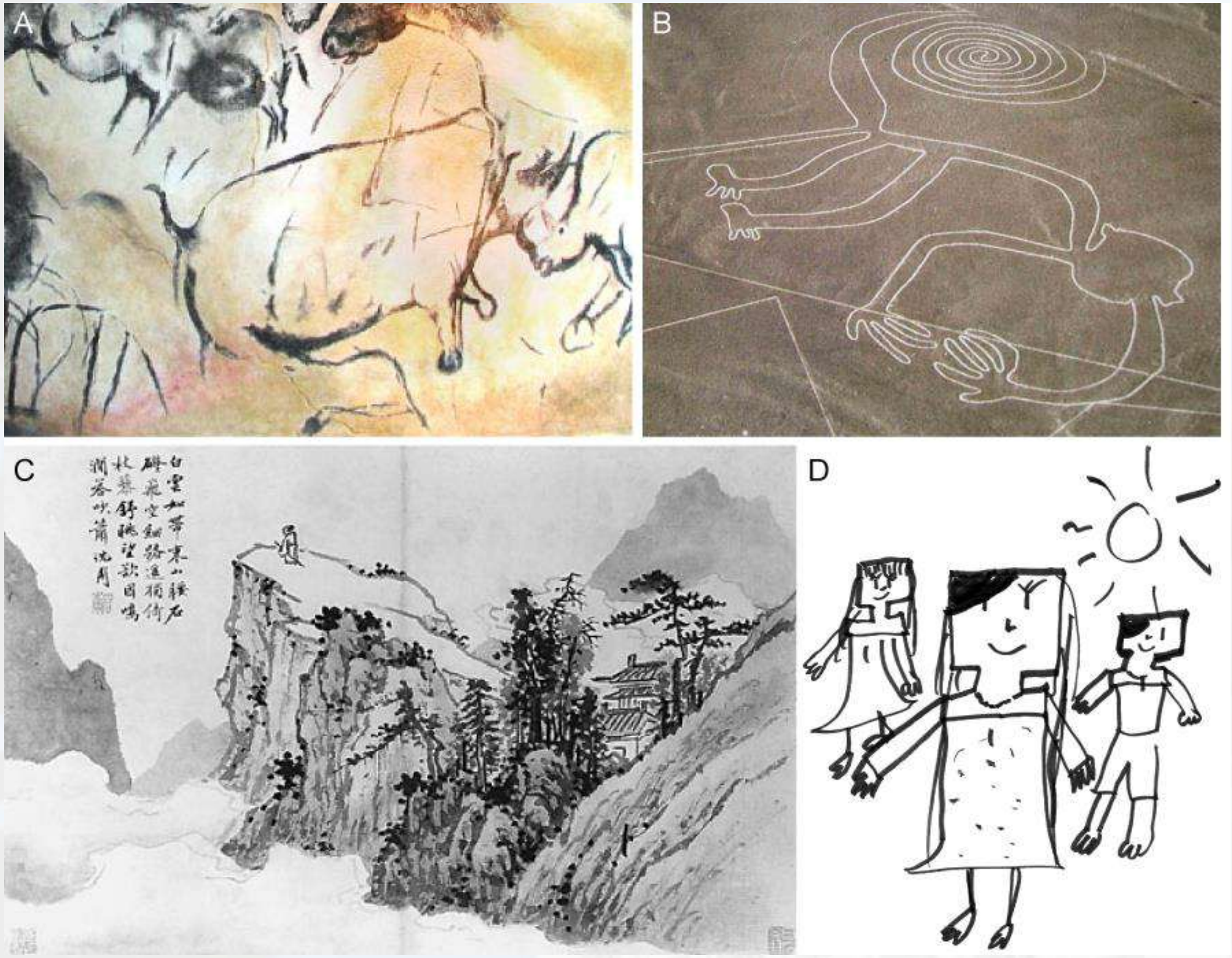
Outline

- Edge Detection Overview
- Image Gradients
- Effects of Noise
- Classical Edge Detectors

Edges

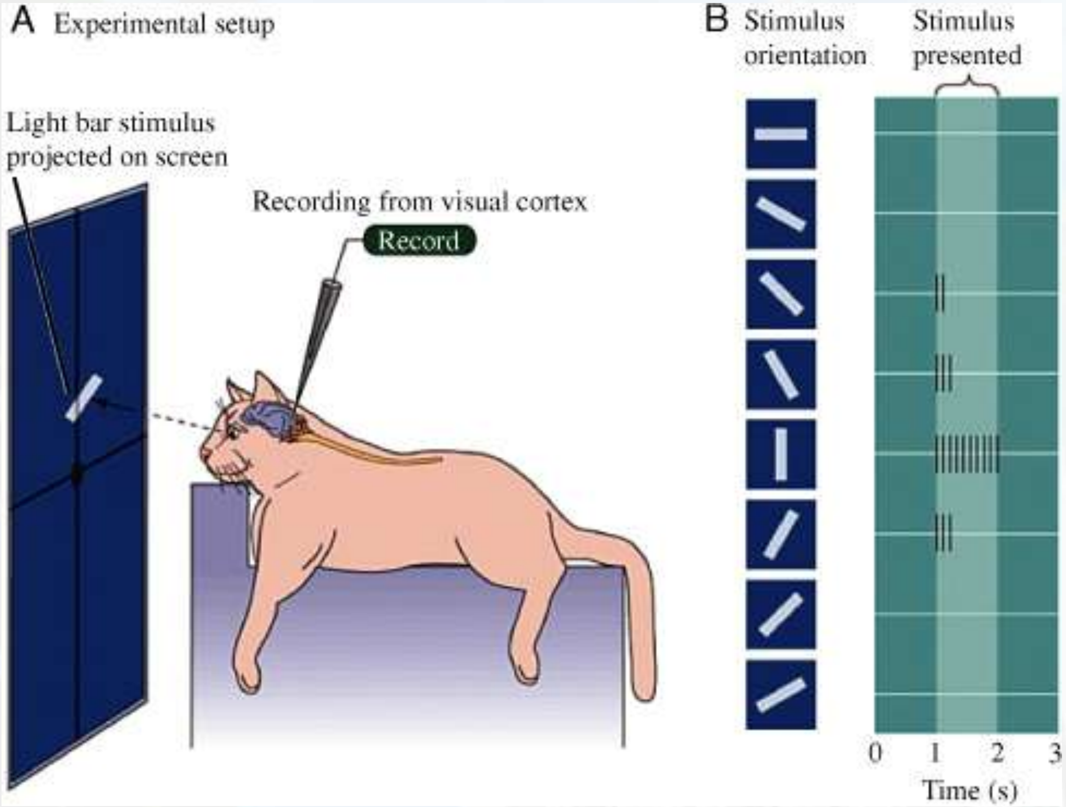


Edges



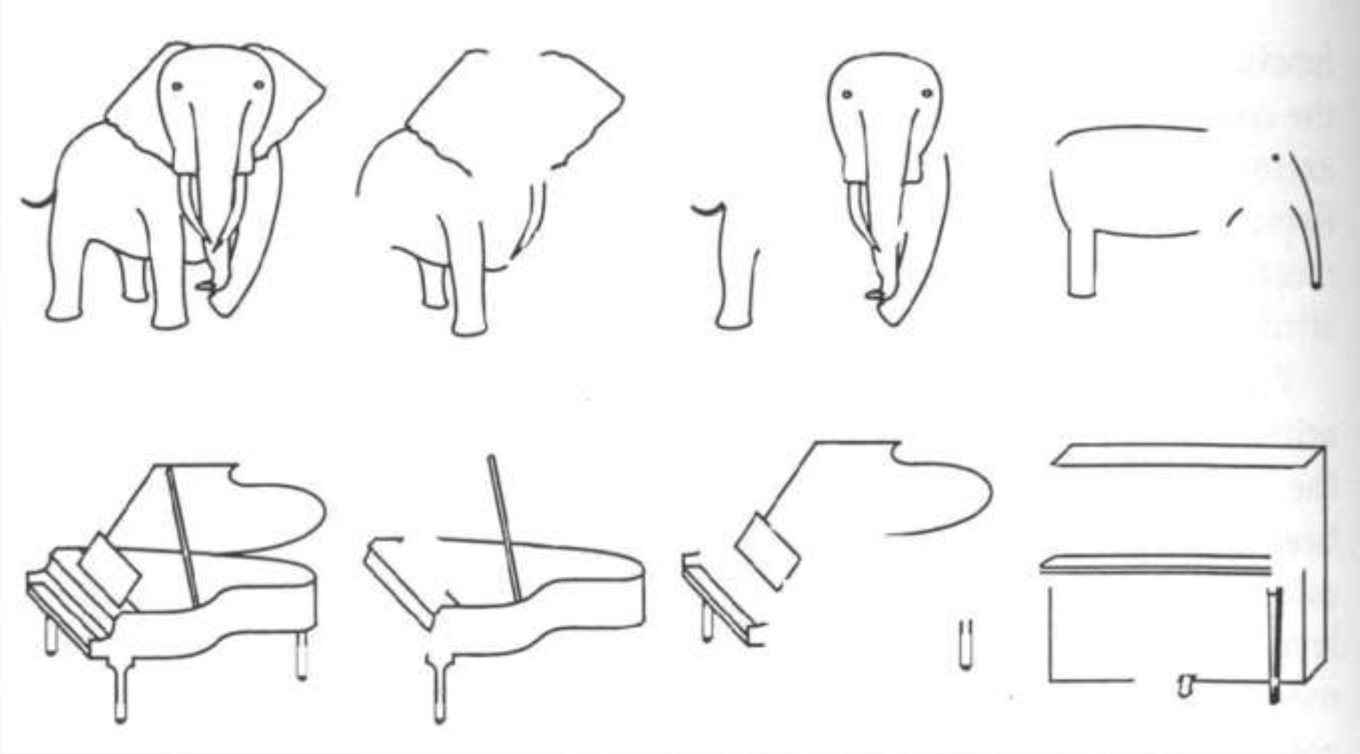
Edge Detection in Mammals

Hubel &Wiesel, 1960



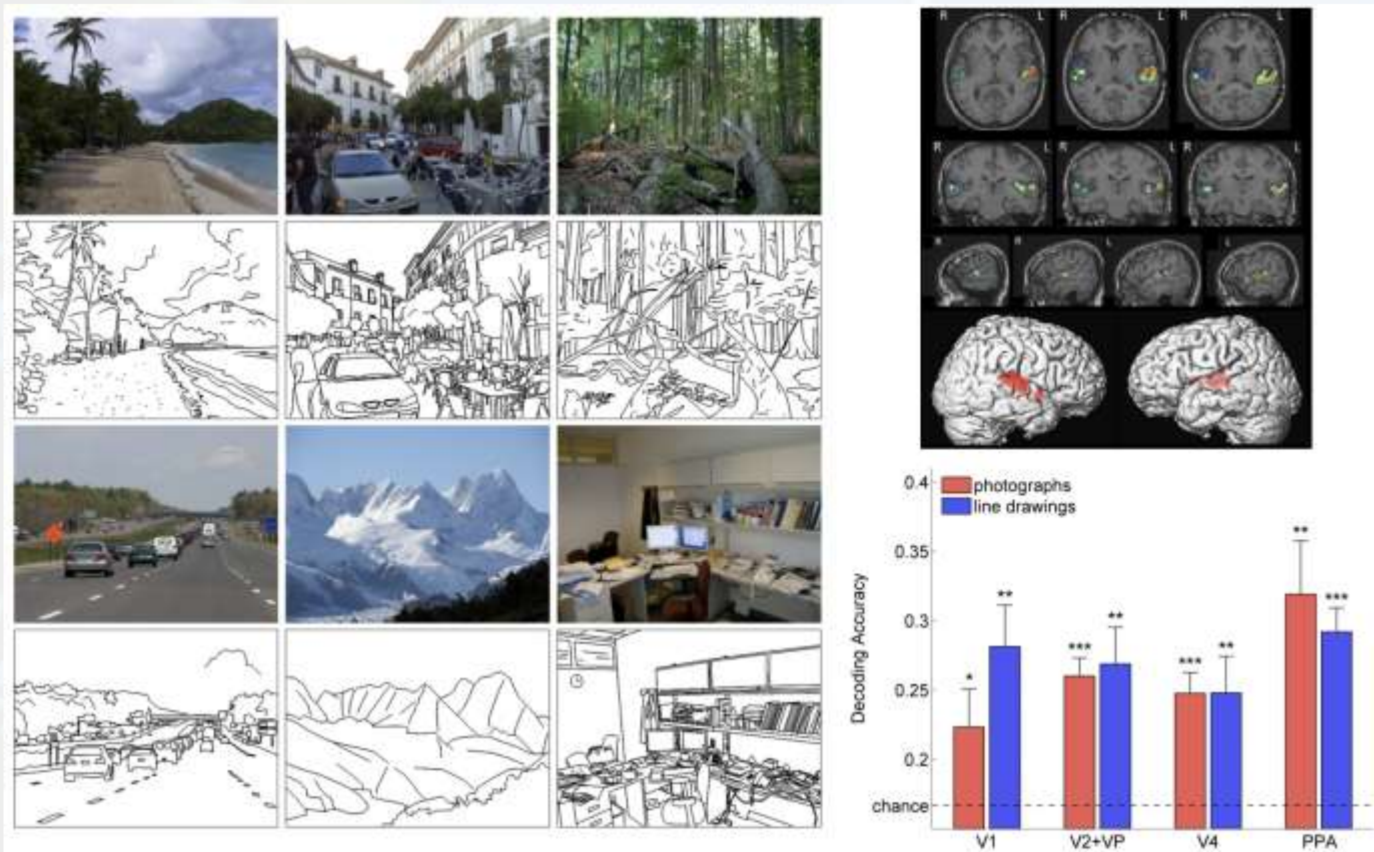
Edge Detection in Mammals

Biederman



Edge Detection in Mammals

Fei-Fei Li etc.



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

• Why do we care about edges?

– Extract information, recognize objects

– Recover geometry and viewpoint

Origin of Edges

surface normal discontinuity

depth discontinuity

surface color discontinuity

illumination discontinuity

Outline

- Edge Detection Overview
- Image Gradients
- Effects of Noise
- Classical Edge Detectors

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 Derivative Filters

- Backward [0 1 -1] $$\frac{df}{dx} = f(x) - f(x-1) = f'(x)$$
- Forward [-1 1 0] $$\frac{df}{dx} = f(x) - f(x+1) = f'(x)$$
- Central [1 0 -1] $$\frac{df}{dx} = f(x+1) - f(x-1) = f'(x)$$

Derivatives in 2D

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 $\theta = tan^{-1}\left(\frac{\frac{df}{dy}}{\frac{df}{dx}}\right)$

2D Discrete Derivative Filters

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

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

2D Discrete Derivative Filters

Derivative mask

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

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

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

2D Discrete Derivative Filters

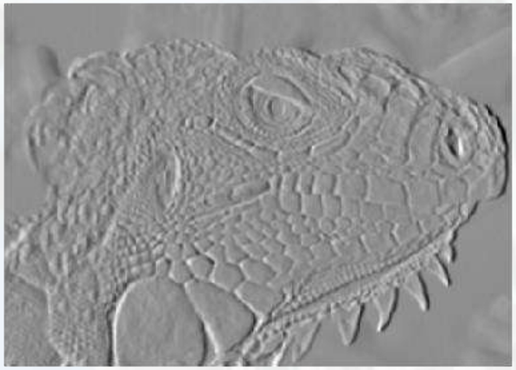
Derivative mask $\frac{1}{3}\begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix}$

$$I = \begin{bmatrix} 10 & 10 & 20 & 20 & 20 \\ 10 & 10 & 20 & 20 & 20 \\ 10 & 10 & 20 & 20 & 20 \\ 10 & 10 & 20 & 20 & 20 \\ 10 & 10 & 20 & 20 & 20 \end{bmatrix} \qquad I_y = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

3x3 image gradient filters

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

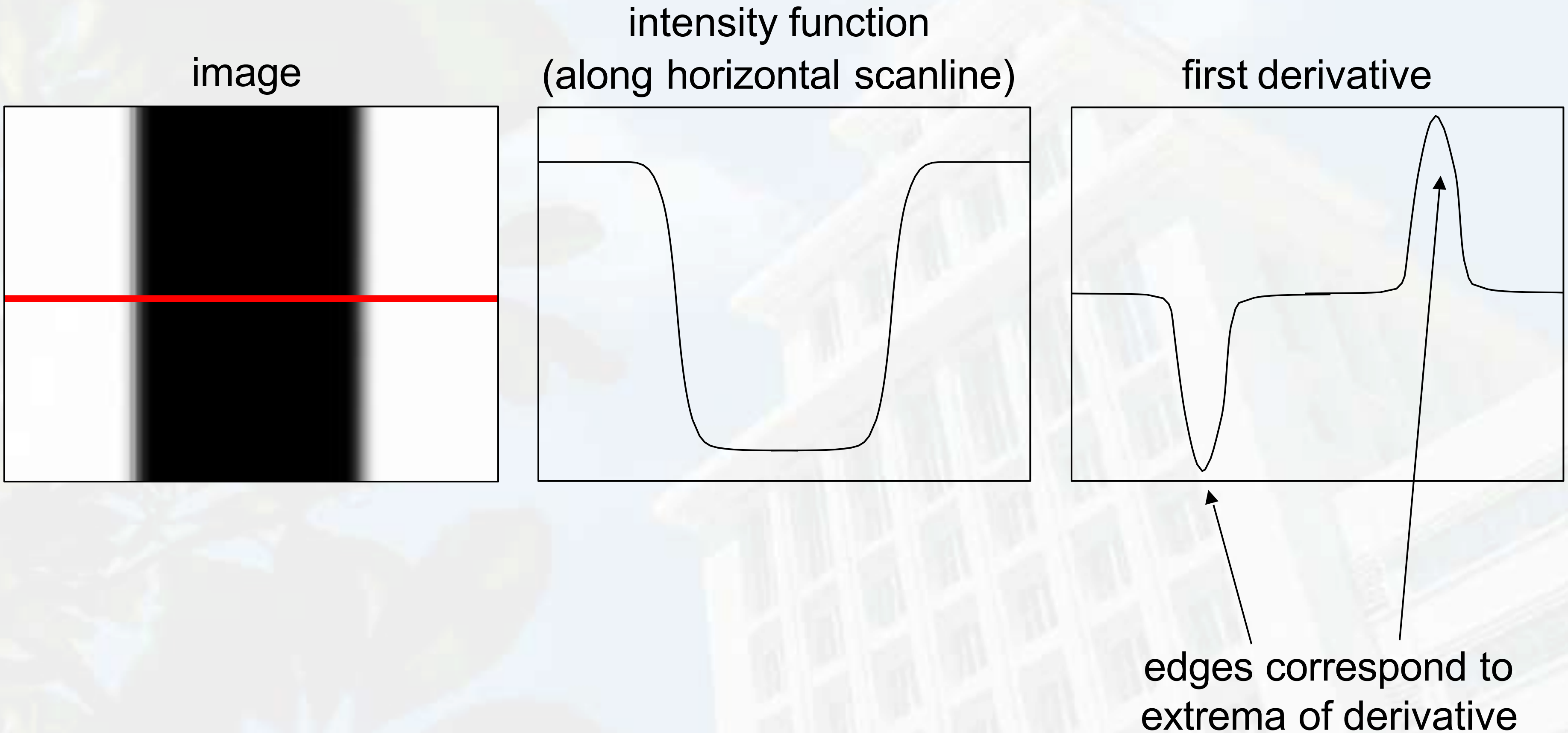
Derivative in x direction



Derivative in y direction

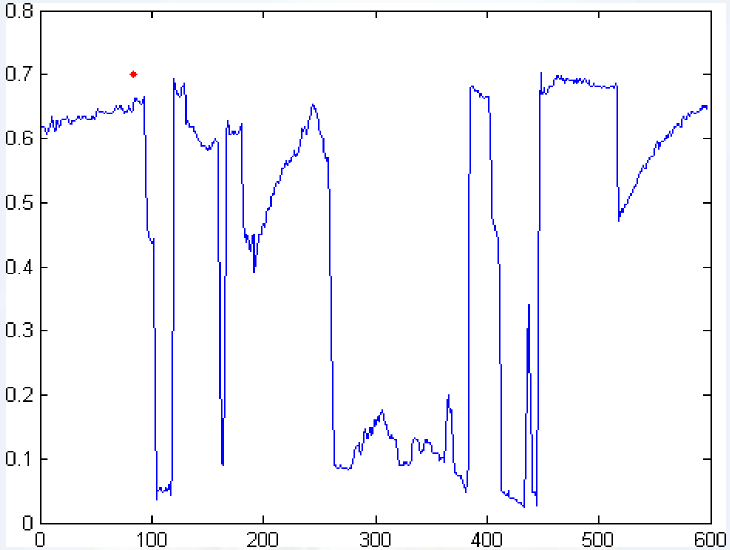
Characterizing edges

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



Intensity profile

intensity



gradient

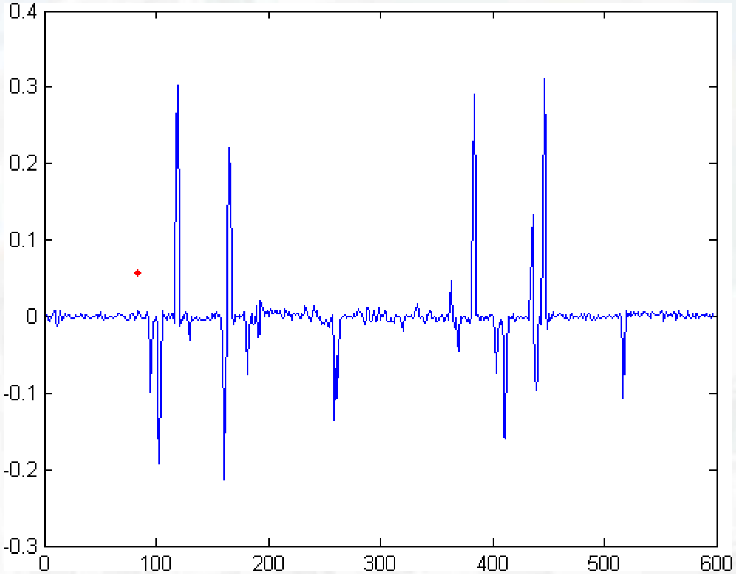
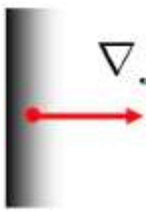


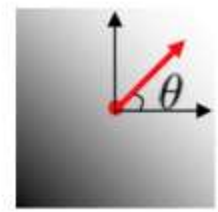
Image gradient

- 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

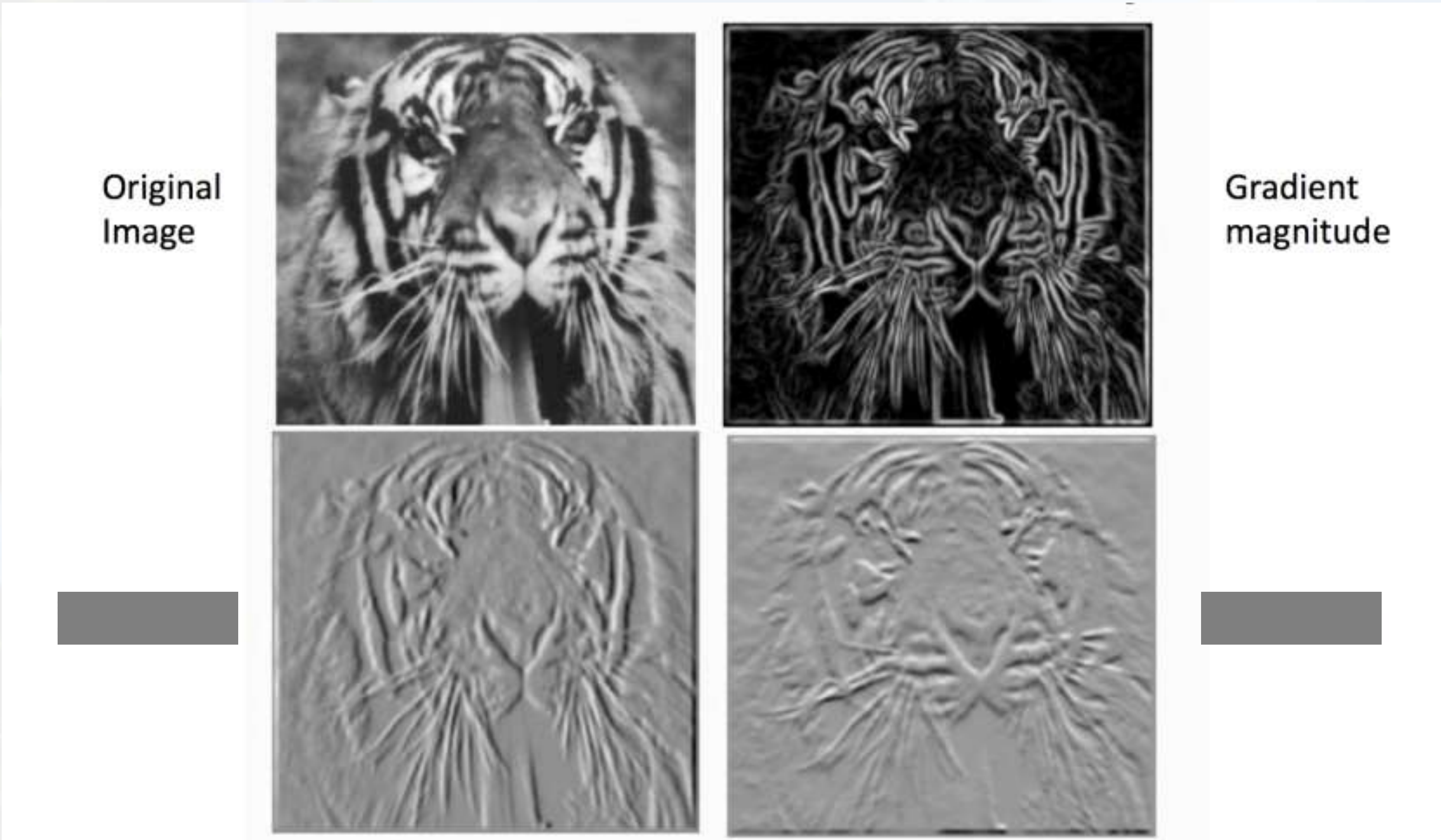
The gradient direction is given by $\theta = \tan^{-1}\left(\frac{\partial f}{\partial y} / \frac{\partial f}{\partial x}\right)$

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

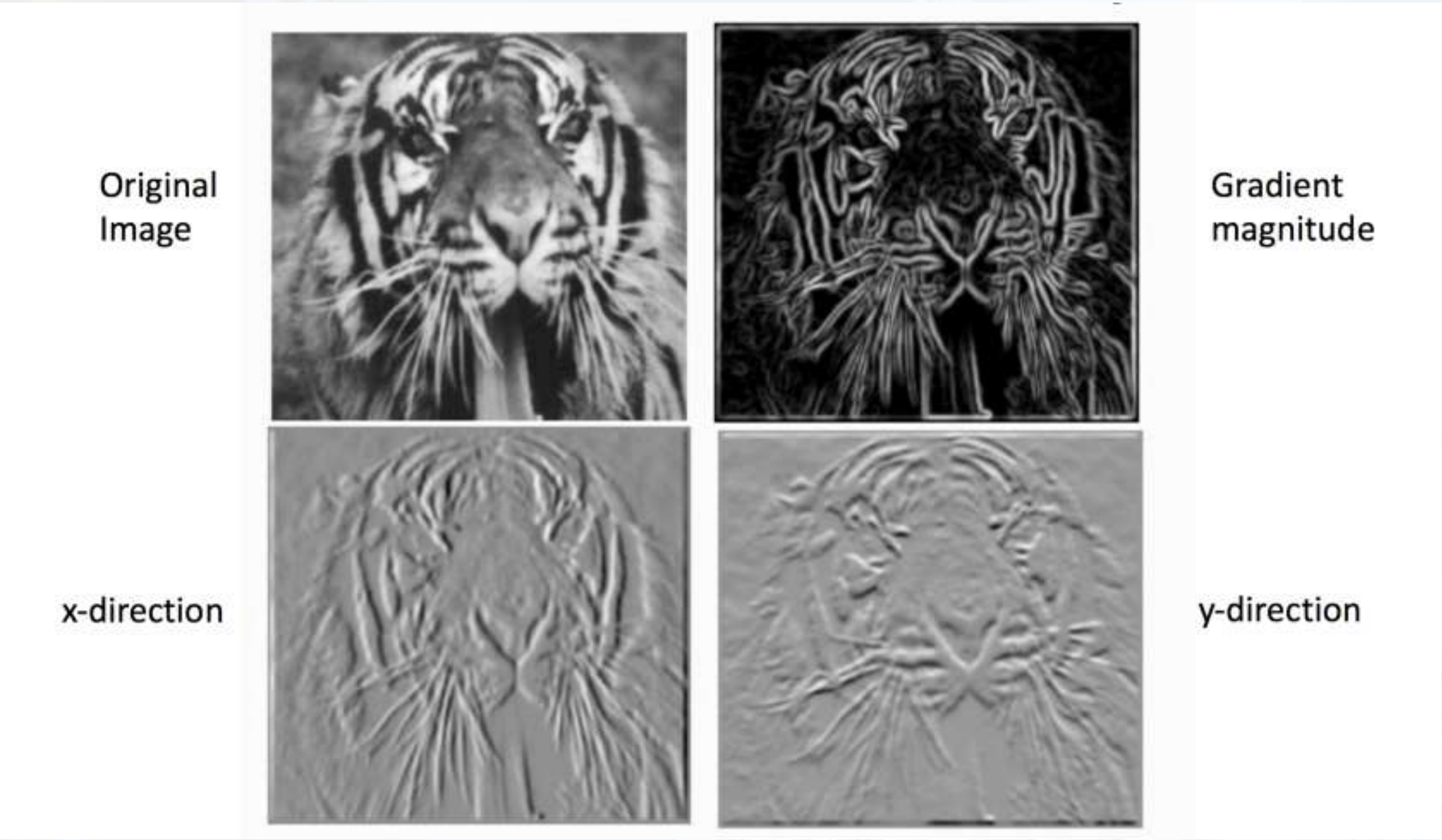
The *edge strength* is given by the gradient magnitude

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

Discrete derivative/gradient: example



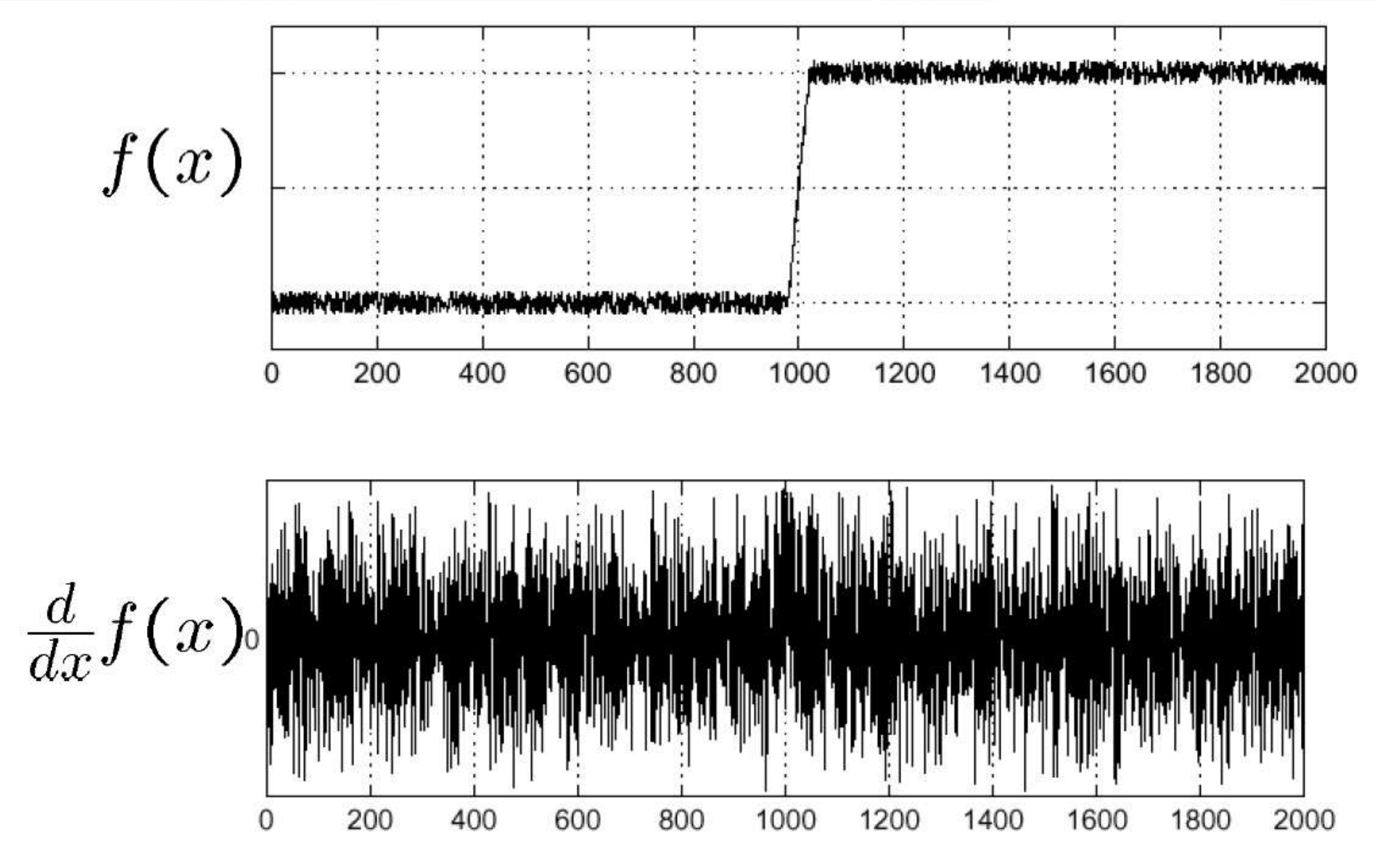
Discrete derivative/gradient: example



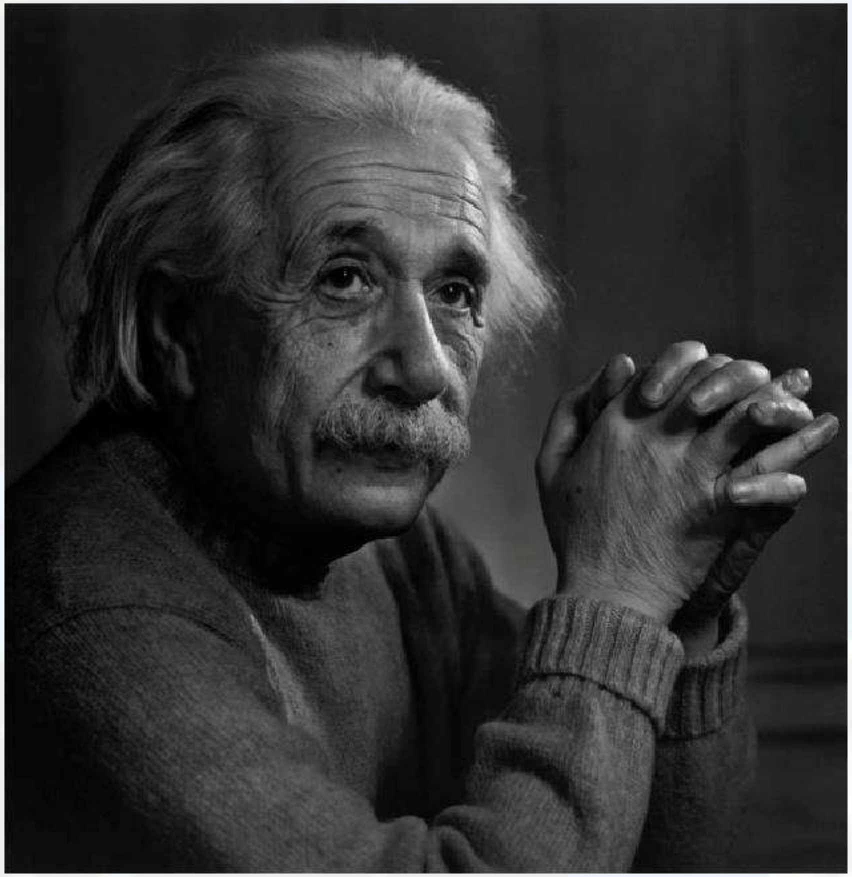
Outline

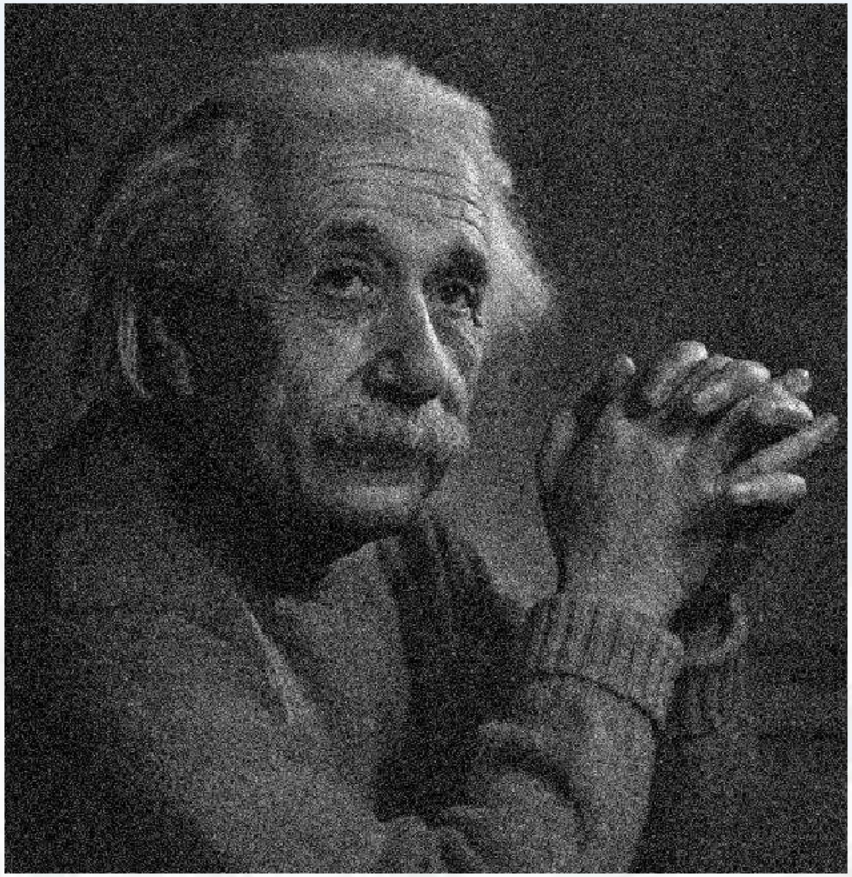
- Edge Detection Overview
- Image Gradients
- Effects of Noise
- Classical Edge Detectors

Effects of Noise



Effects of Noise





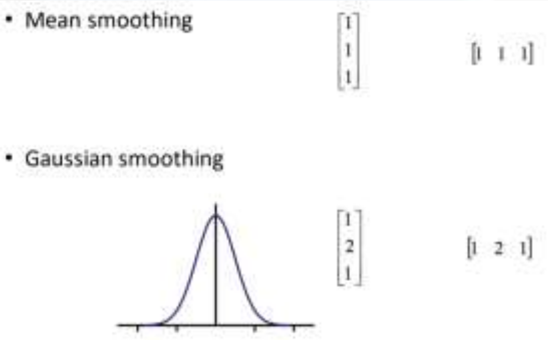
Effects of Noise

- Discrete gradient 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

Effects of Noise

•Discrete gradient 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



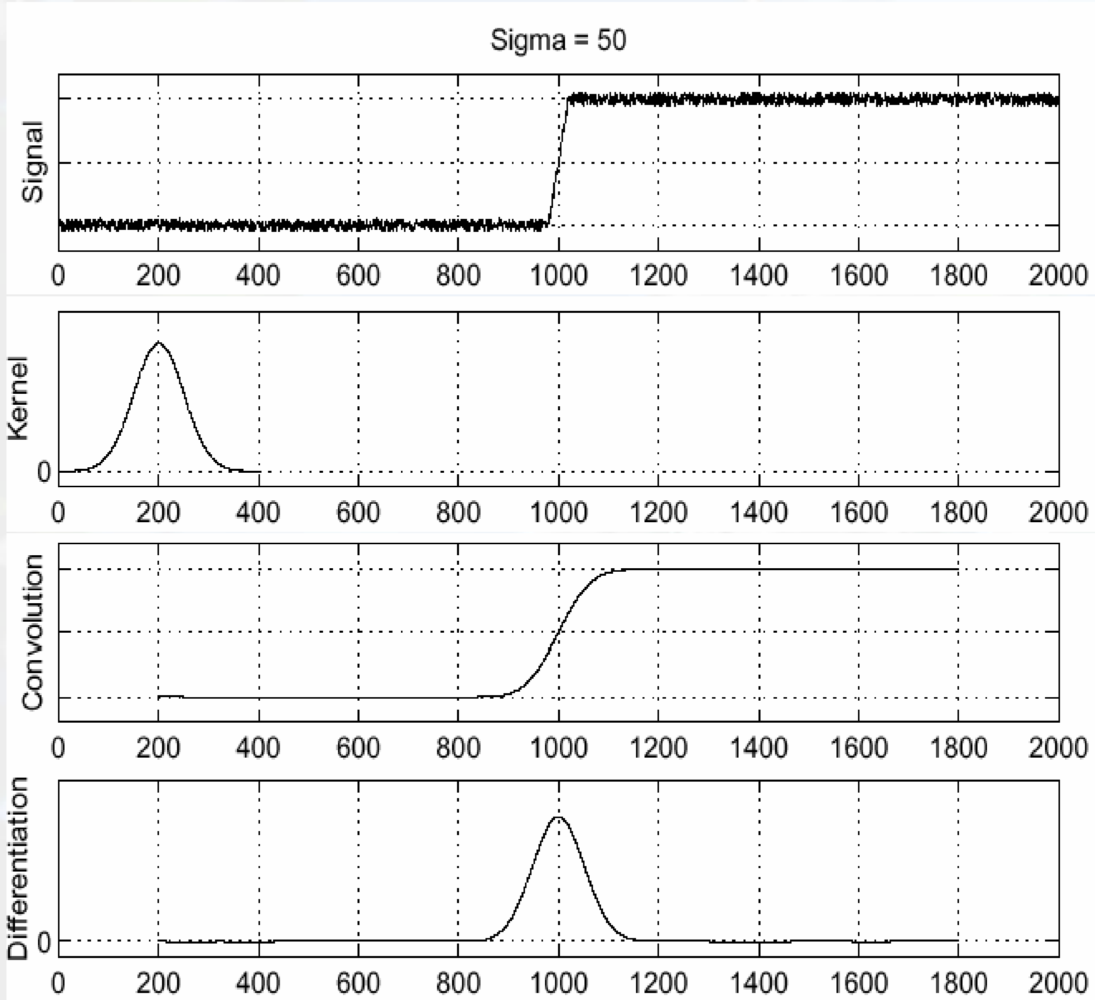
Solution: Smooth First

f

g

$f * g$

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

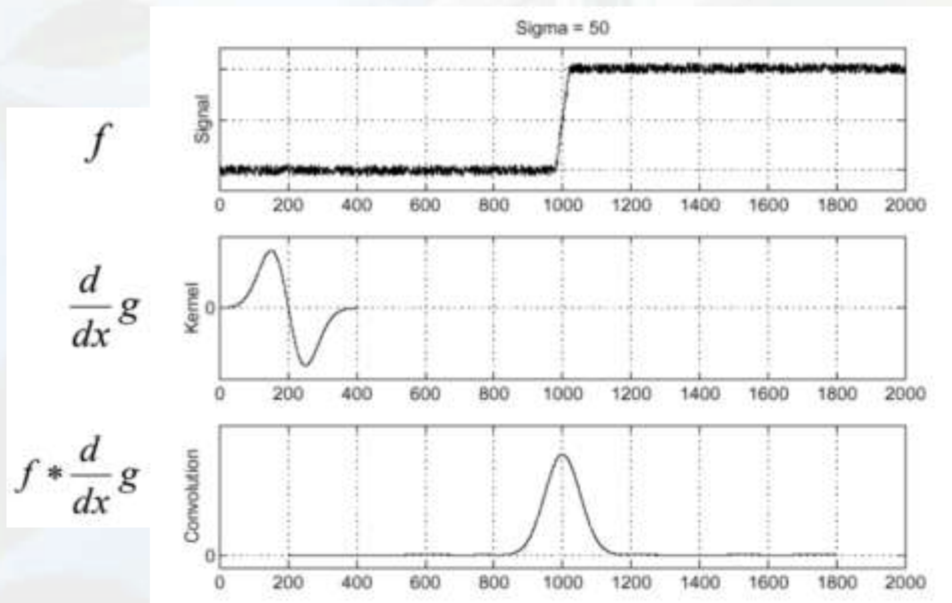


Derivative Theorem of Convolution

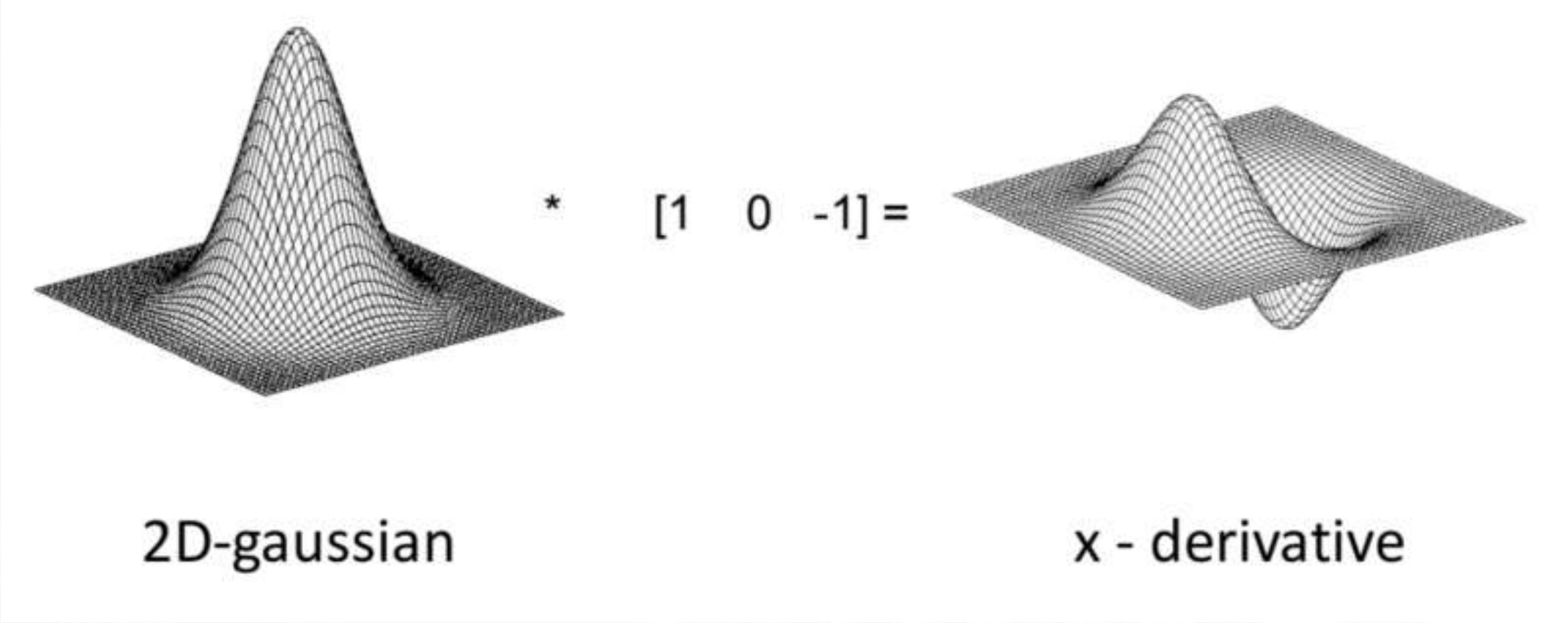
- This theorem gives a very useful property:

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

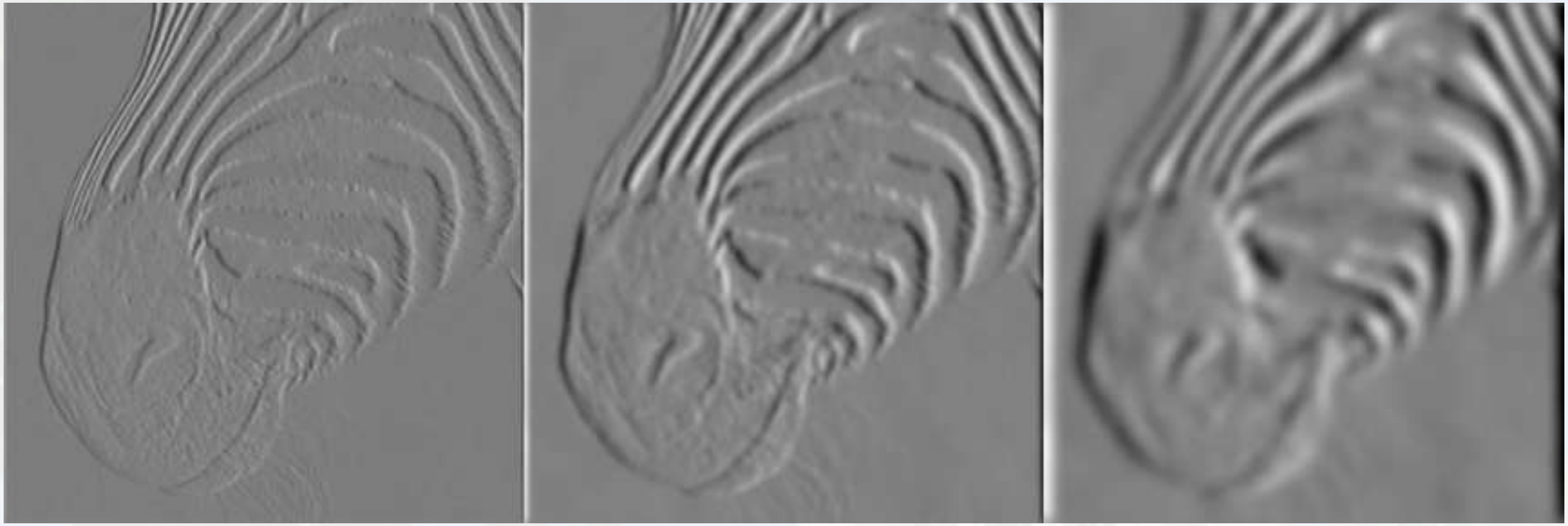
- This saves us one operation



Derivative of Gaussian Filter



Tradeoff between smoothing and localization



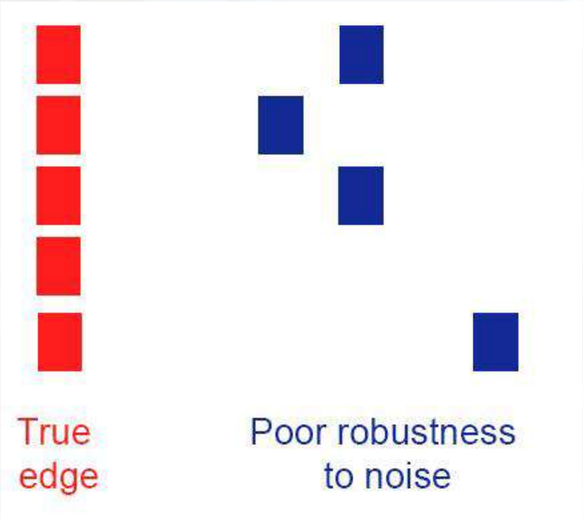
1 pixel (Kernel size) 3 pixels 7 pixels

- Stronger smoothing removes noise, but blurs edge.
- Finds edges at different "scales".

Design an Edge Detector

- Criteria for an "optimal" edge detector:

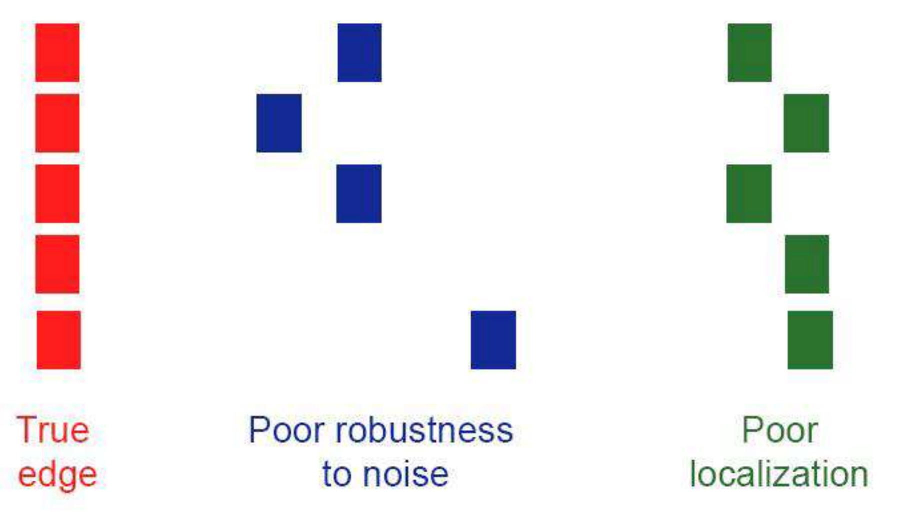
–**Good detection**: the optimal detector must minimize the probability of false positives (detecting spurious edges caused by noise), as well as that of false negatives (missing real edges)



Design an Edge Detector

•Criteria for an “optimal” edge detector:

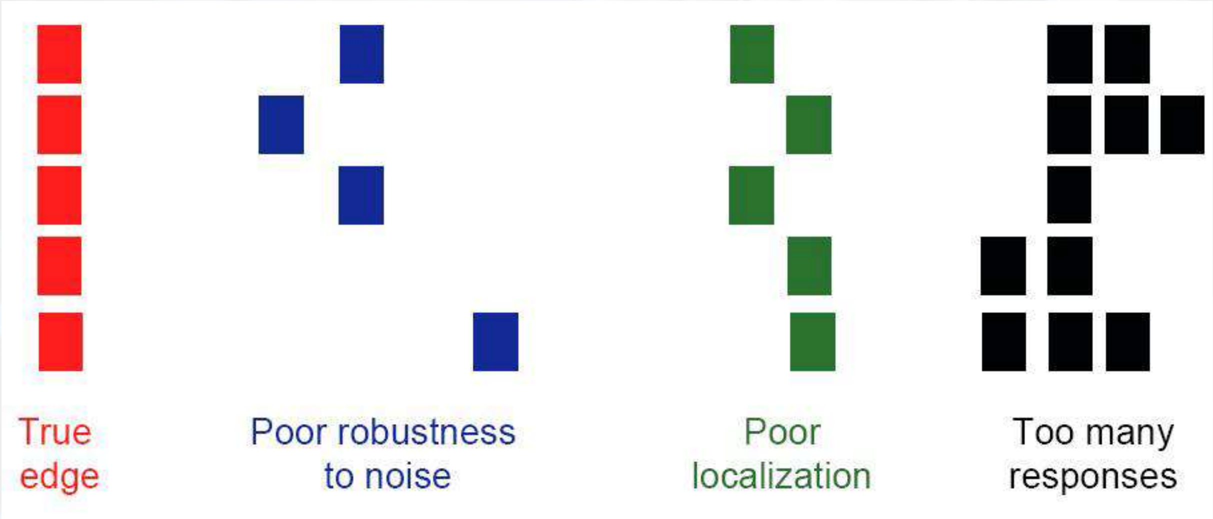
–**Good detection**: the optimal detector must minimize the probability of false positives (detecting spurious edges caused by noise), as well as that of false negatives (missing real edges)
–**Good localization:** the edges detected must be as close as possible to the true edges



Design 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



Outline

- Edge Detection Overview
- Image Gradients
- Effects of Noise
- Classical Edge Detectors

Classical Detectors

- Gradient operators – Sobel
- Laplacian of Gaussian - LoG
- Gradient of Gaussian (Canny)

Sobel Edge Detector

- Named after Irwin Sobel and Gary Feldman, colleagues at the Stanford Artificial Intelligence Laboratory (SAIL). Sobel and Feldman presented the idea of an ”Isotropic 3 × 3 Image Gradient Operator" at a talk at SAIL in 1968.

- 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

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

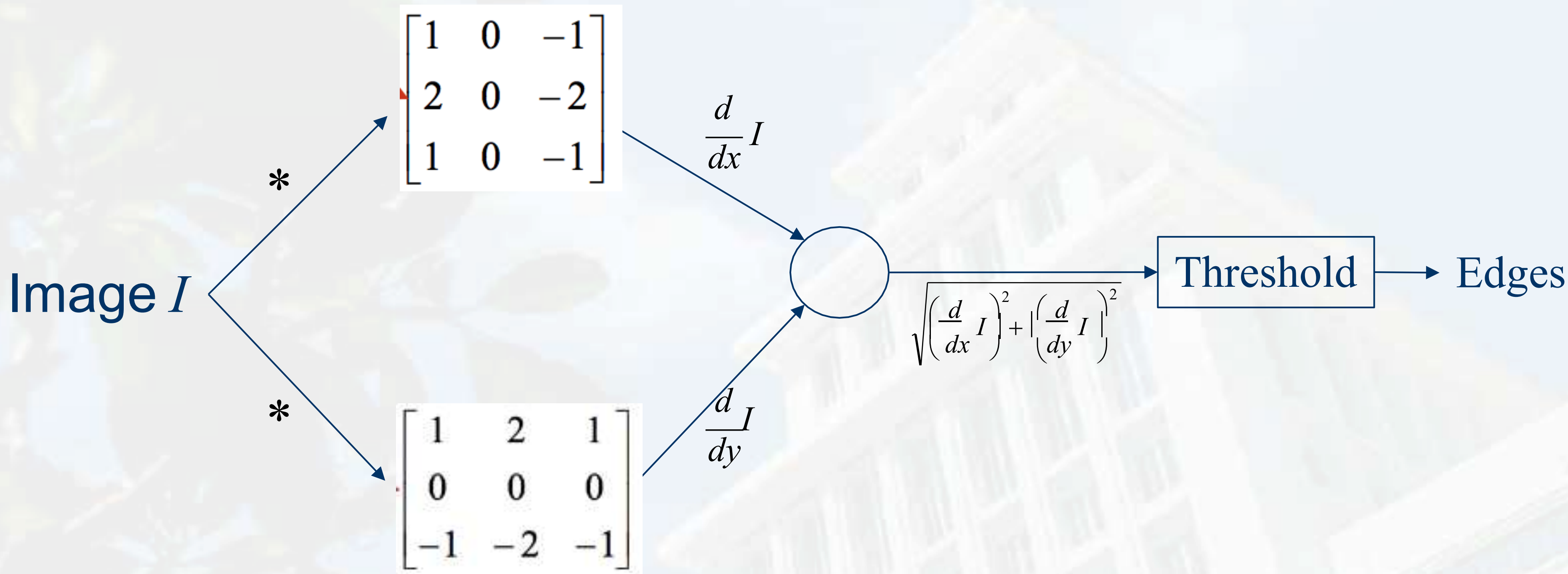
$$\begin{bmatrix} +1 & 0 & -1 \\ +2 & 0 & -2 \\ +1 & 0 & -1 \end{bmatrix} = \underbrace{\begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix}}_{\text{Gaussian smoothing}} \underbrace{\begin{bmatrix} +1 & 0 & -1 \end{bmatrix}}_{\text{differentiation}}$$

Sobel Edge Detector

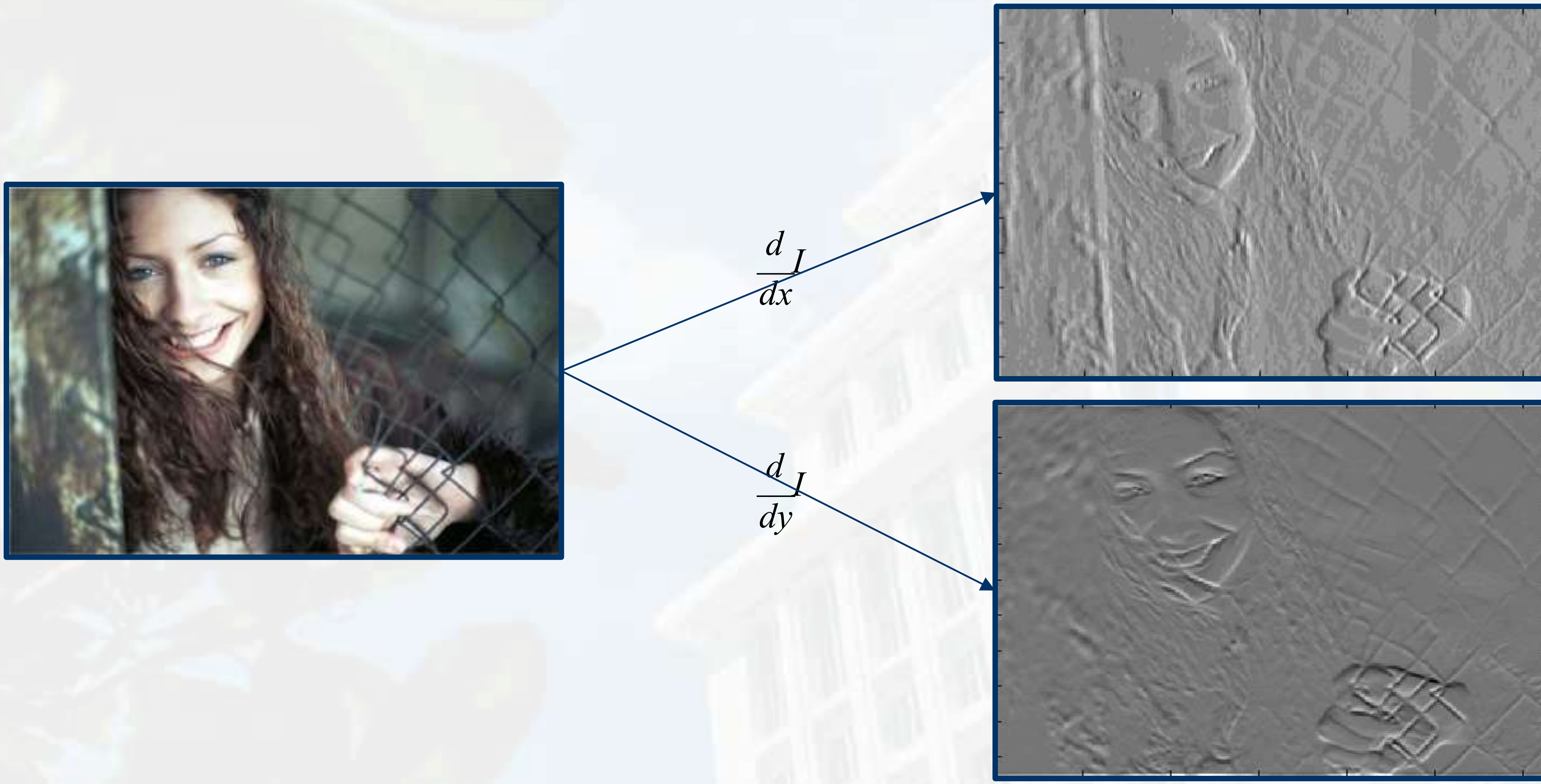
SETPS:

1. Compute derivatives
   - In x and y directions
2. Find gradient magnitude
3. Threshold gradient magnitude

Sobel Edge Detector



Sobel Edge Detector

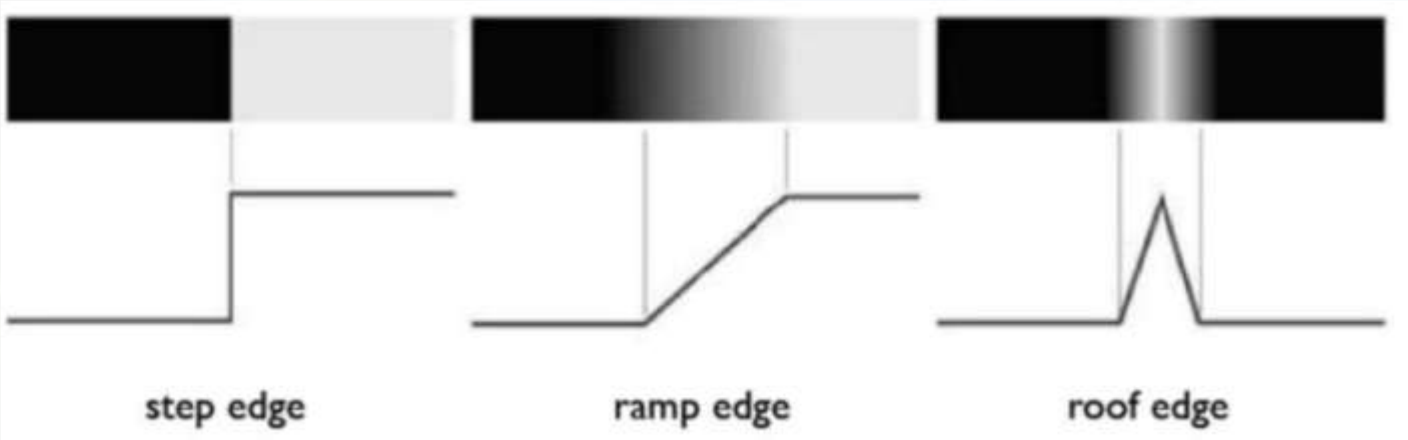


Sobel Edge Detector

$$\Delta = \sqrt{\left(\frac{d}{dx}I\right)^2 + \left(\frac{d}{dy}I\right)^2}$$

$$\Delta \geq Threshold = 100$$

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

Classical Detectors

- Gradient operators – Sobel
- Laplacian of Gaussian - LoG
- Gradient of Gaussian (Canny)

Marr Hildreth Edge Detector

Proceedings of the Royal Society of London. Series B. Biological Sciences
rspb.royalsocietypublishing.org

Published 29 February 1980 doi: 10.1098/rspb.1980.0020
Proc. R. Soc. Lond. B **29 February 1980** vol. 207 no. 1167 **187-217**

Theory of Edge Detection

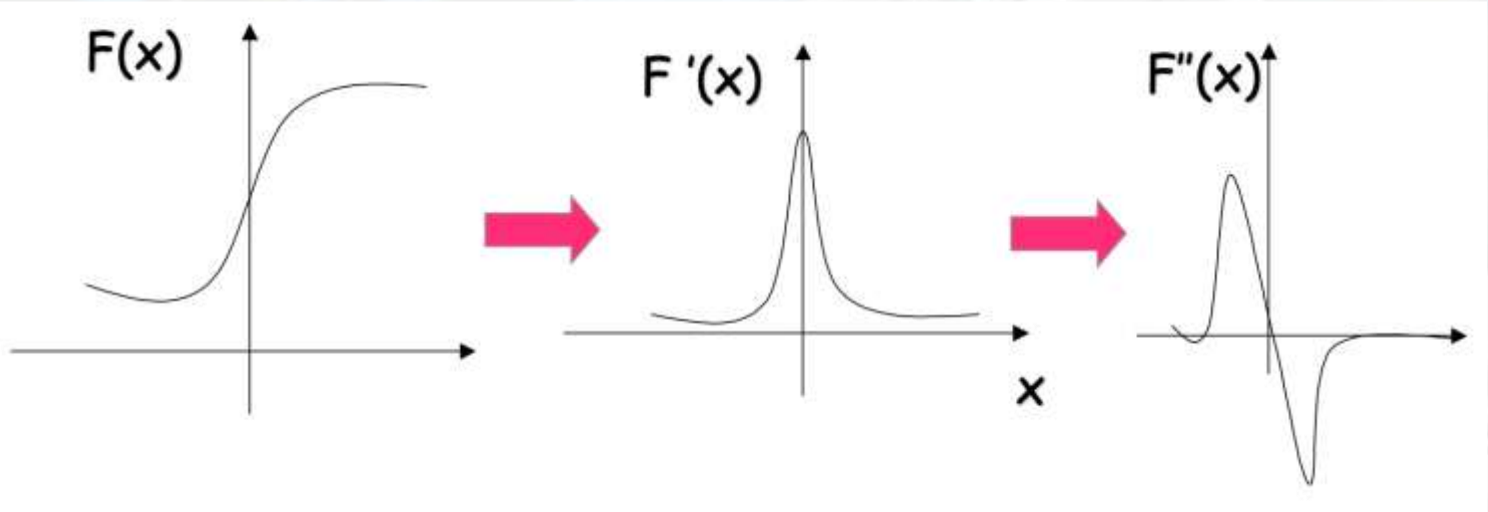
D. Marr and E. Hildreth

Abstract

A theory of edge detection is presented. The analysis proceeds in two parts. (1) Intensity changes, which occur in a natural image over a wide range of scales, are detected separately at different scales. An appropriate filter for this purpose at a given scale is found to be the second derivative of a Gaussian, and it is shown that, provided some simple conditions are satisfied, these primary filters need not be orientation-dependent. Thus, intensity changes at a given scale are best detected by finding the zero values of $\nabla ^{2}$G(x, y)* I(x, y) for image I, where G(x, y) is a two-dimensional Gaussian distribution and $\nabla ^{2}$ is the Laplacian. The intensity changes thus discovered in each of the channels are then represented by oriented primitives called zero-crossing segments, and evidence is given that this representation is complete. (2) Intensity changes in images arise from surface discontinuities or from reflectance or illumination boundaries, and these all have the property that they are spatially localized. Because of this, the zero-crossing segments from the different channels are not independent, and rules are deduced for combining them into a description of the image. This description is called the raw primal sketch. The theory explains several basic

Marr Hildreth Edge Detector

Peaks or valleys of the first-derivative of the input signal, correspond to “zero-crossings” of the second-derivative of the input signal.



Marr Hildreth Edge Detector

- Find edges by second order differentiation
- Use the Gaussian smoothing operator to improve the response to noise, and by differentiation the Laplacian of Gaussian is called the LoG operator.
- Edges are at the 'zero crossings' of the LoG, which is where there is a change in gradient.

Marr Hildreth Edge Detector

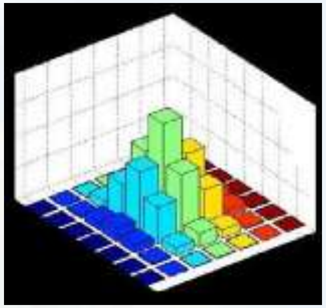
STEPS:

1. Smooth image by Gaussian filter ➔ S
2. Apply Laplacian to S
3. Find zero crossing s

Marr Hildreth Edge Detector

- Gaussian smoothing

$$g = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{x^2+y^2}{2\sigma^2}}$$



- Find Laplacian

$$\Delta^2 S = \frac{\partial^2}{\partial x^2} S + \frac{\partial^2}{\partial y^2} S$$

- ∇ is used for gradient (first derivative)
- Δ^2 is used for Laplacian (Second derivative)

Marr Hildreth Edge Detector

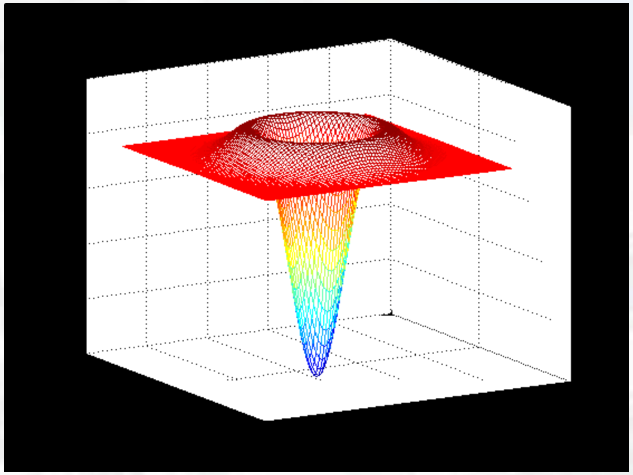
- Deriving the Laplacian of Gaussian (LoG)

$$\Delta^2 S = \Delta^2 (g * I) = (\Delta^2 g) * I$$

$$g = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

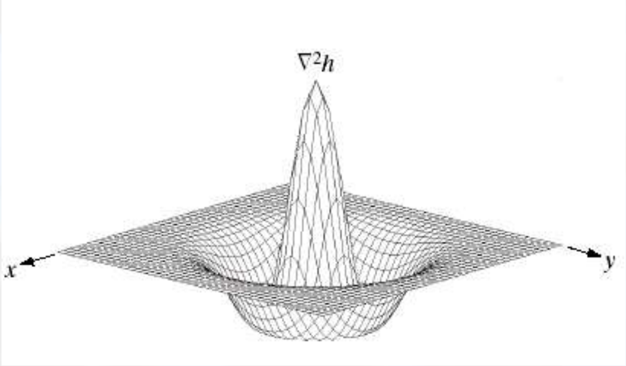
$$g_x = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{x^2+y^2}{2\sigma^2}} \left(-\frac{2x}{2\sigma^2} \right)$$

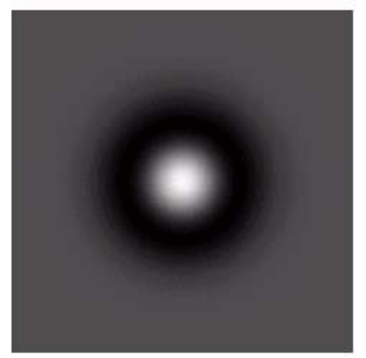
$$\Delta^2 g = -\frac{1}{\sqrt{2\pi}\sigma^3} \left(2 - \frac{x^2+y^2}{\sigma^2} \right) e^{-\frac{x^2+y^2}{2\sigma^2}}$$



LoG Filter

The Laplacian of Gaussian (LoG, or Mexican hat) filter uses the Gaussian for noise removal and the Laplacian for edge detection





0	0	−1	0	0
0	−1	−2	−1	0
−1	−2	16	−2	−1
0	−1	−2	−1	0
0	0	−1	0	0

Find Zero-crossing

- Perform 8 neighbor zero crossing, check pixel signs in 4 directions
- 4 directions: left-right, up-down, 2 diagonals
- At least 2 opposite signs in 4 directions
- Typically, not only opposite signs are required, but also the difference needs surpass certain threshold

On the Separability of LoG

- Two-dimensional Gaussian can be separated into 2 one-dimensional Gaussians

$$h(x, y) = I(x, y) * g(x, y)$$ n^2 multiplications

$$h(x, y) = \left(I(x, y) * g_1(x)\right) * g_2(y)$$ $2n$ multiplications

$$\mathrm{g}(x) = e^{-\left(\frac{x^2}{2\sigma^2}\right)}$$

$$g_1 = g(x) = \begin{bmatrix} .011 & .13 & .6 & 1 & .6 & .13 & .011 \end{bmatrix}$$

$$\mathrm{g}_2 = g(y) = \begin{bmatrix} .011 \\ .13 \\ .6 \\ 1 \\ .6 \\ .13 \\ .011 \end{bmatrix}$$

On the Separability of LoG

Similar to separability of Gaussian filter, two-dimensional LoG can be separated into 4 one-dimensional Convolutions

$$\Delta^2 S = \Delta^2 (g * I) = (\Delta^2 g) * I = I * (\Delta^2 g)$$

Requires n^2 multiplications

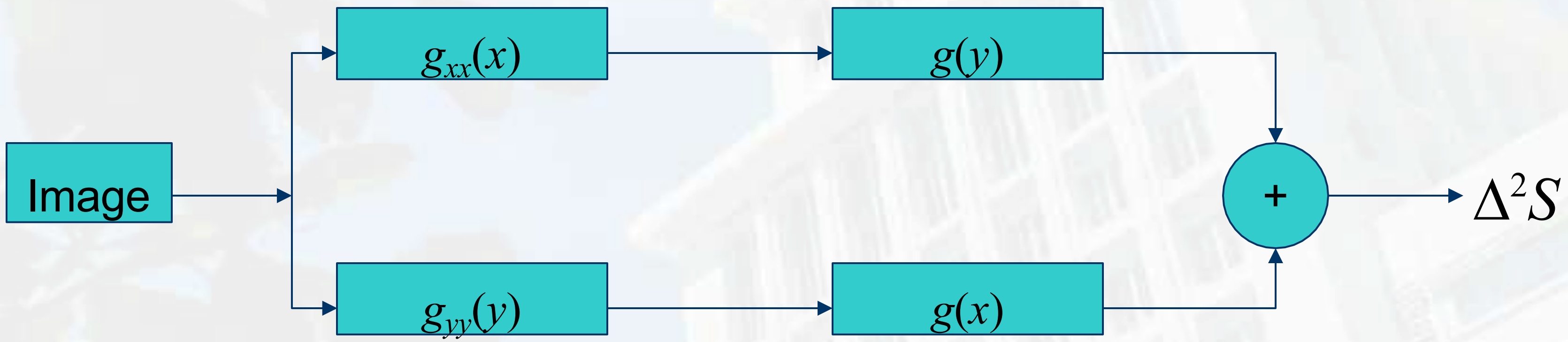
$$\Delta^2 S = (I * g_{xx}(x)) * g(y) + (I * g_{yy}(y)) * g(x)$$

Requires $4n$ multiplications

Seperability

Gaussian Filtering

Laplacian of Gaussian Filtering

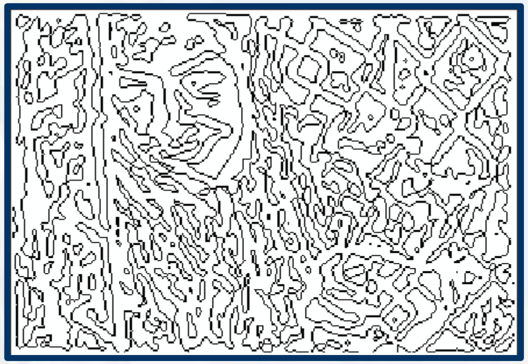


Example

I

$I * \left(\Delta^2 g\right)$

Zero crossings of $\Delta^2 S$



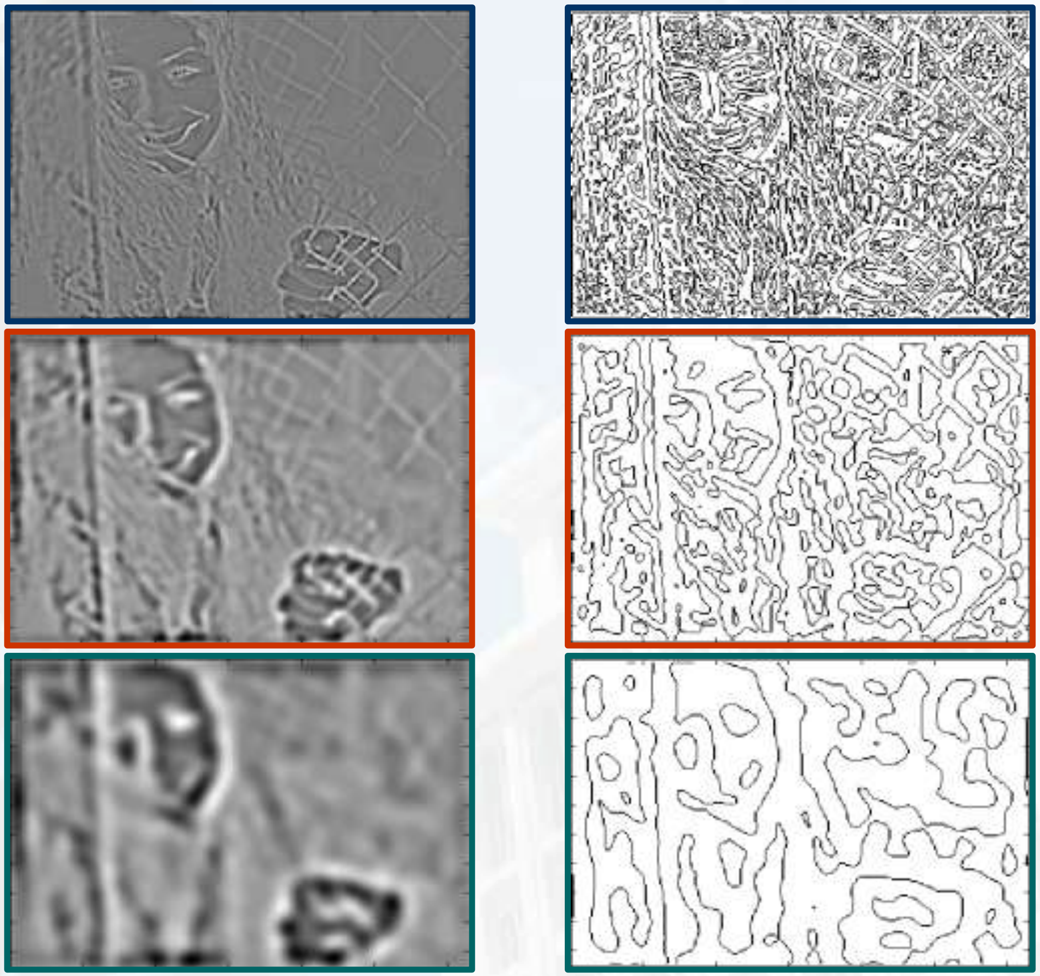
$$\Delta^2 G_\sigma = -\frac{1}{\sqrt{2\pi}\sigma^3}\left(2-\frac{x^2+y^2}{\sigma^2}\right)e^{-\left(\frac{x^2+y^2}{2\sigma^2}\right)}$$

Example

$\sigma = 1$

$\sigma = 3$

$\sigma = 6$



Classical Detectors

- Gradient operators – Sobel
- Laplacian of Gaussian - LoG
- Gradient of Gaussian (Canny)

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 Steps

1. Smooth image with Gaussian filter
2. Compute derivative of filtered image
3. Find magnitude and orientation of gradient
4. Apply “Non-maximum Suppression”
5. Apply “Hysteresis Threshold”

Canny Edge Detector
First Two Steps

- Smoothing

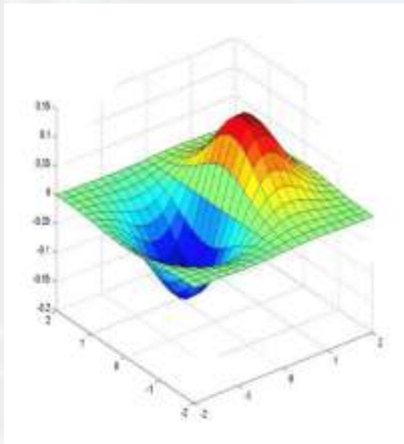
$$S = I * g(x,y) = g(x,y) * I \qquad g(x,y) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

- Derivative

$$\nabla S = \nabla(g * I) = (\nabla g) * I \qquad \nabla g = \begin{bmatrix} \frac{\partial g}{\partial x} \\ \frac{\partial g}{\partial y} \end{bmatrix} = \begin{bmatrix} g_x \\ g_y \end{bmatrix}$$

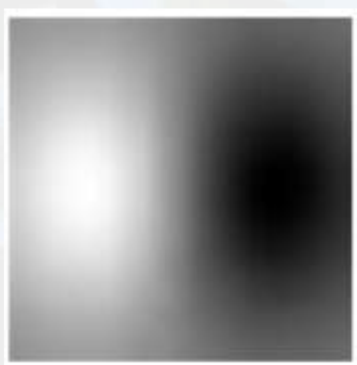
$$\nabla S = \begin{bmatrix} g_x \\ g_y \end{bmatrix} * I = \begin{bmatrix} g_x * I \\ g_y * I \end{bmatrix}$$

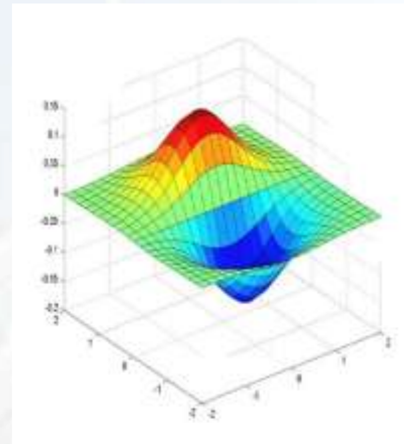
Derivative of Gaussian filter



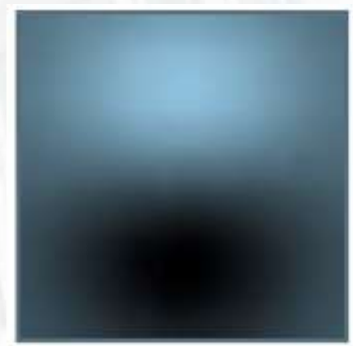
x-direction

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





y-direction



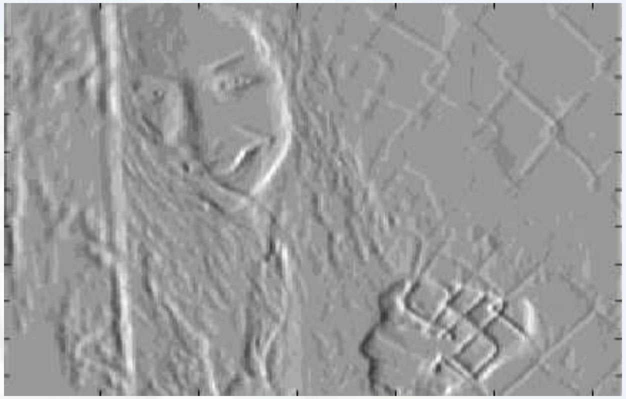
$$\begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

Canny Edge Detector
First Two Steps

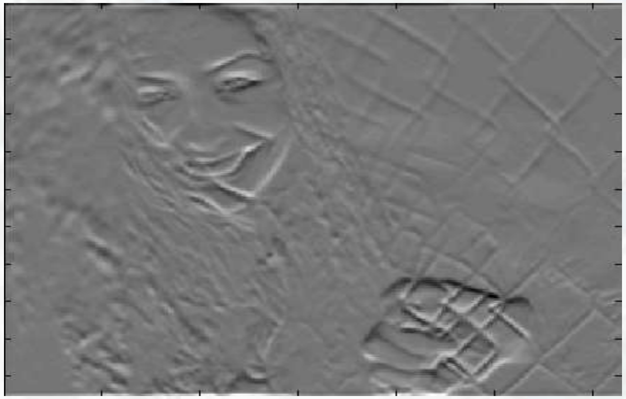
(S_x, S_y) Gradient Vector

I

S_x



S_y



Canny Edge Detector
Third Step

- Gradient magnitude and gradient direction

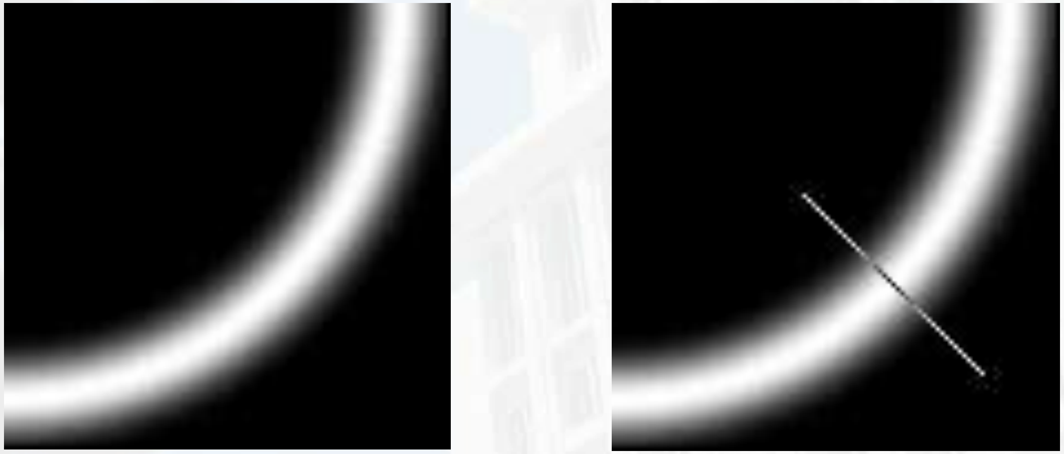
image

gradient magnitude

Canny Edge Detector
Fourth Step

- Non-maximum suppression

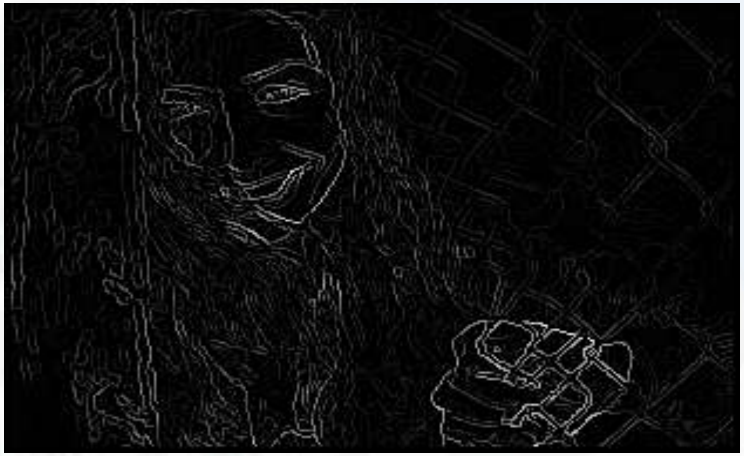
- Edge occurs where gradient reaches a maxima
- Suppress non-maxima gradient even if it passes threshold
- Compare current pixel vs neighbors along direction of gradient
  - Remove if not maximum



Canny Edge Detector Fourth Step

- Non-maximum suppression

Before



After

Canny Edge Detector
Fifth Step

- Hysteresis Thresholding

Detecting edges with a single threshold

Threshold too high

Threshold too low

Canny Edge Detector
Fifth Step

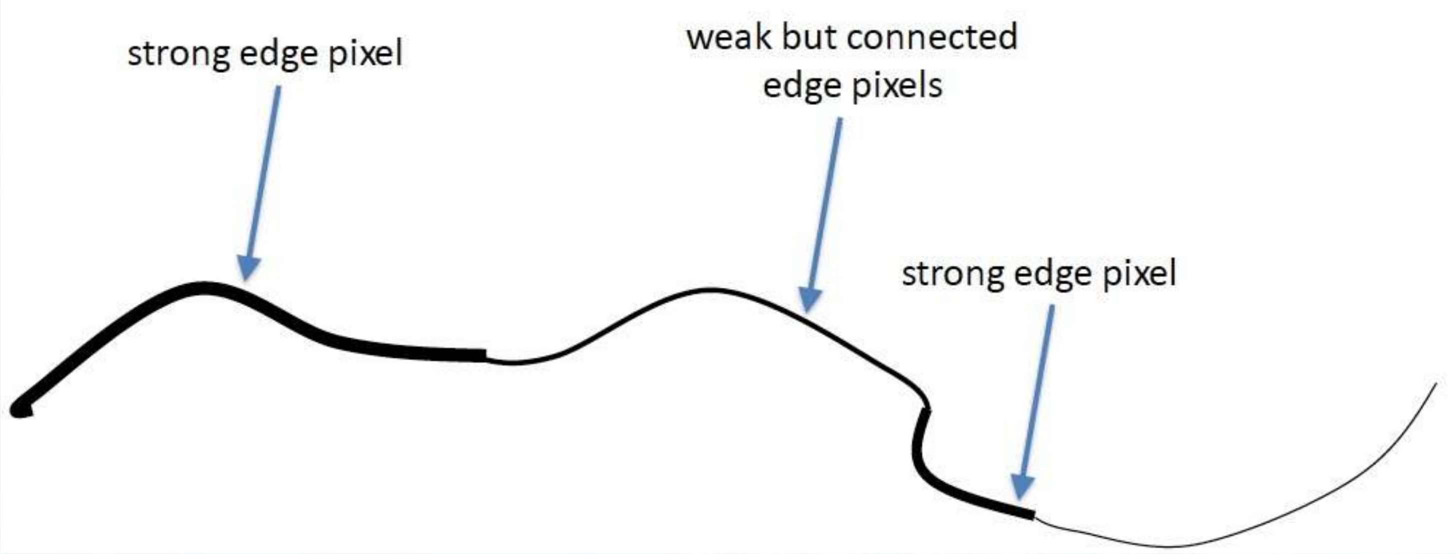
- Hysteresis Thresholding

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”.

Canny Edge Detector
Fifth Step

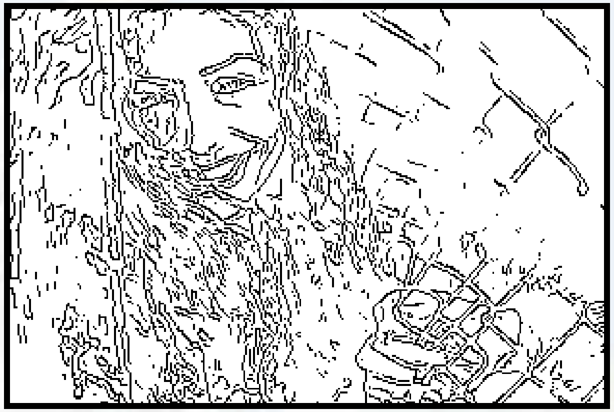
- Hysteresis Thresholding



Canny Edge Detector Fifth Step

- Hysteresis Thresholding

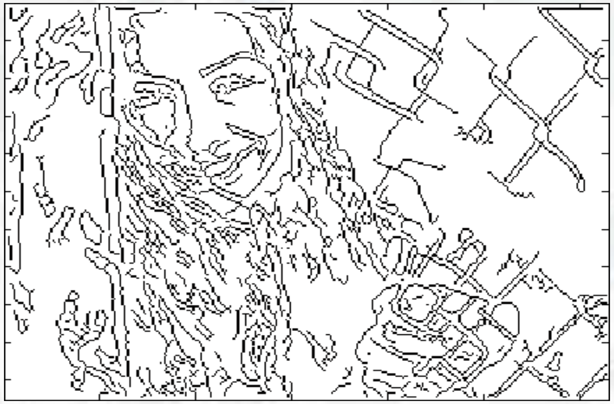
Single threshold =25



Hysteresis

$High = 35$

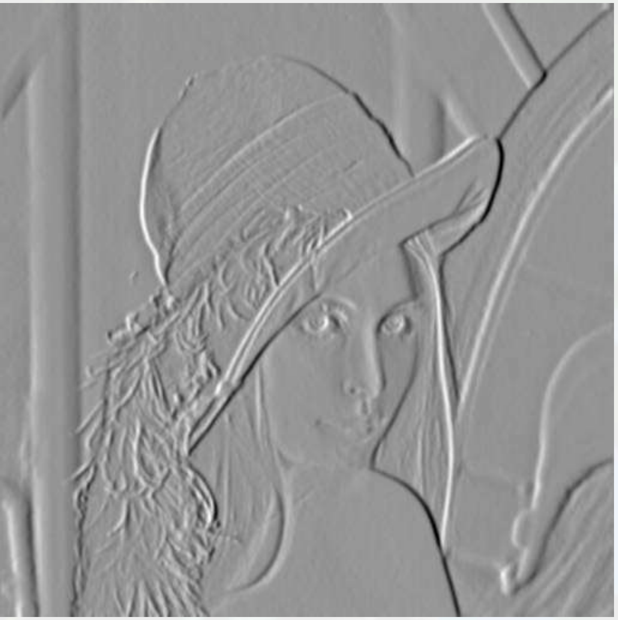
$Low = 15$



Example of Canny Detector

Original image (Lena)

Compute Gradient Derivative

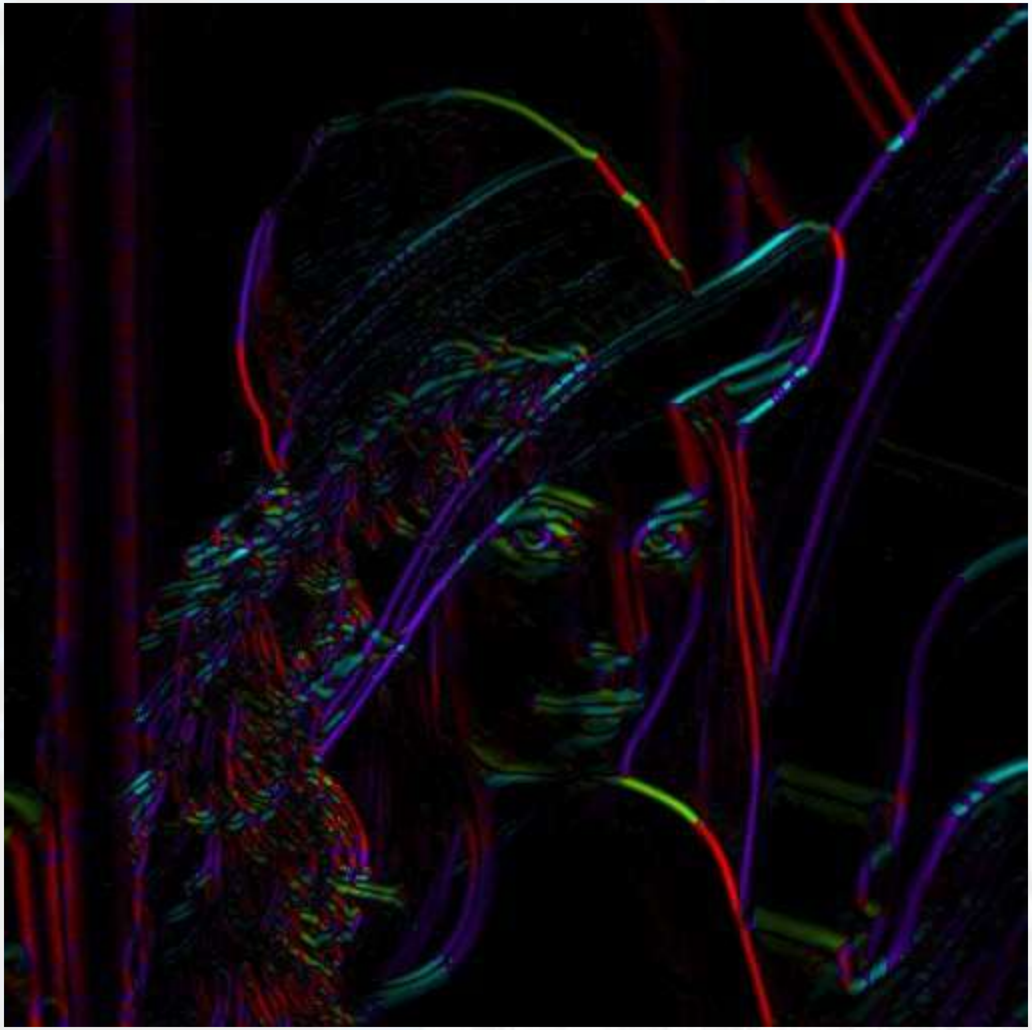


X-Derivative of Gaussian

Y-Derivative of Gaussian

Gradient Magnitude

Get Orientation at Each Pixel



Before Non-max Suppression

After non-max suppression

Hysteresis thresholding

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