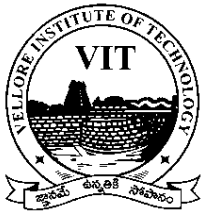


VIT-AP UNIVERSITY

Computer Vision

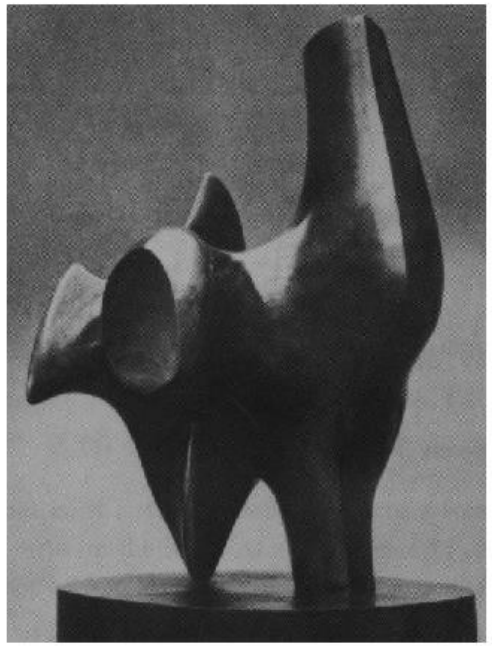
(Course Code: 4047)

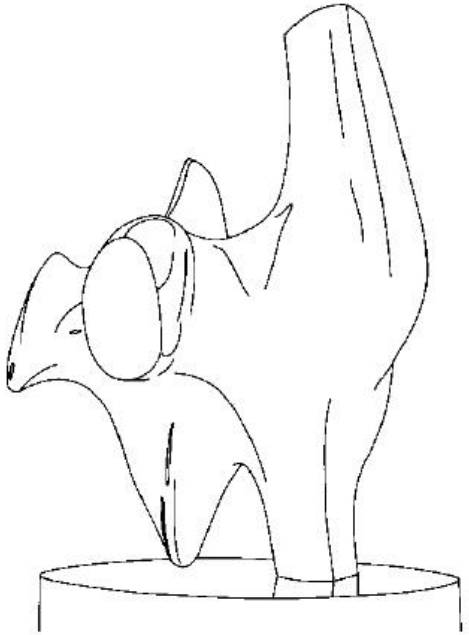
Module-2:Lecture-1: EDGE DETECTION

Gundimeda Venugopal, Professor of Practice, SCOPE

Edge Detection

What is an Edge? Edge is a rapid Change in Image Intensity in a small region

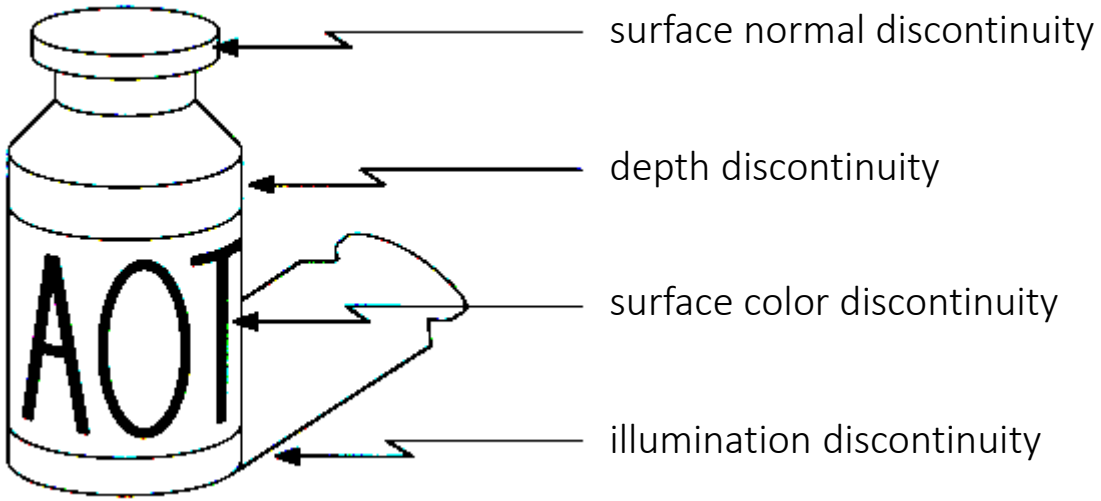




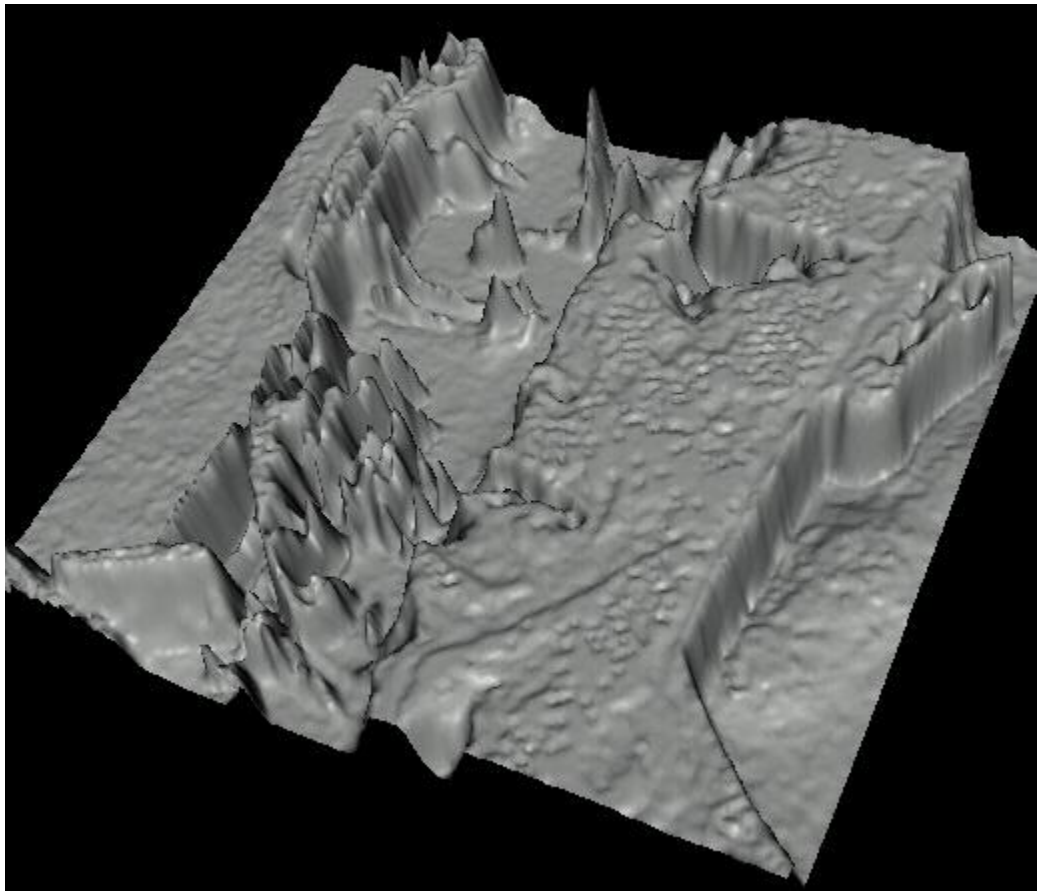
- Convert a 2D image into a set of curves
  - Extracts salient features of the scene
  - More compact than pixels

Origin of edges

- Rapid changes in image intensity are caused by a various physical phenomena.

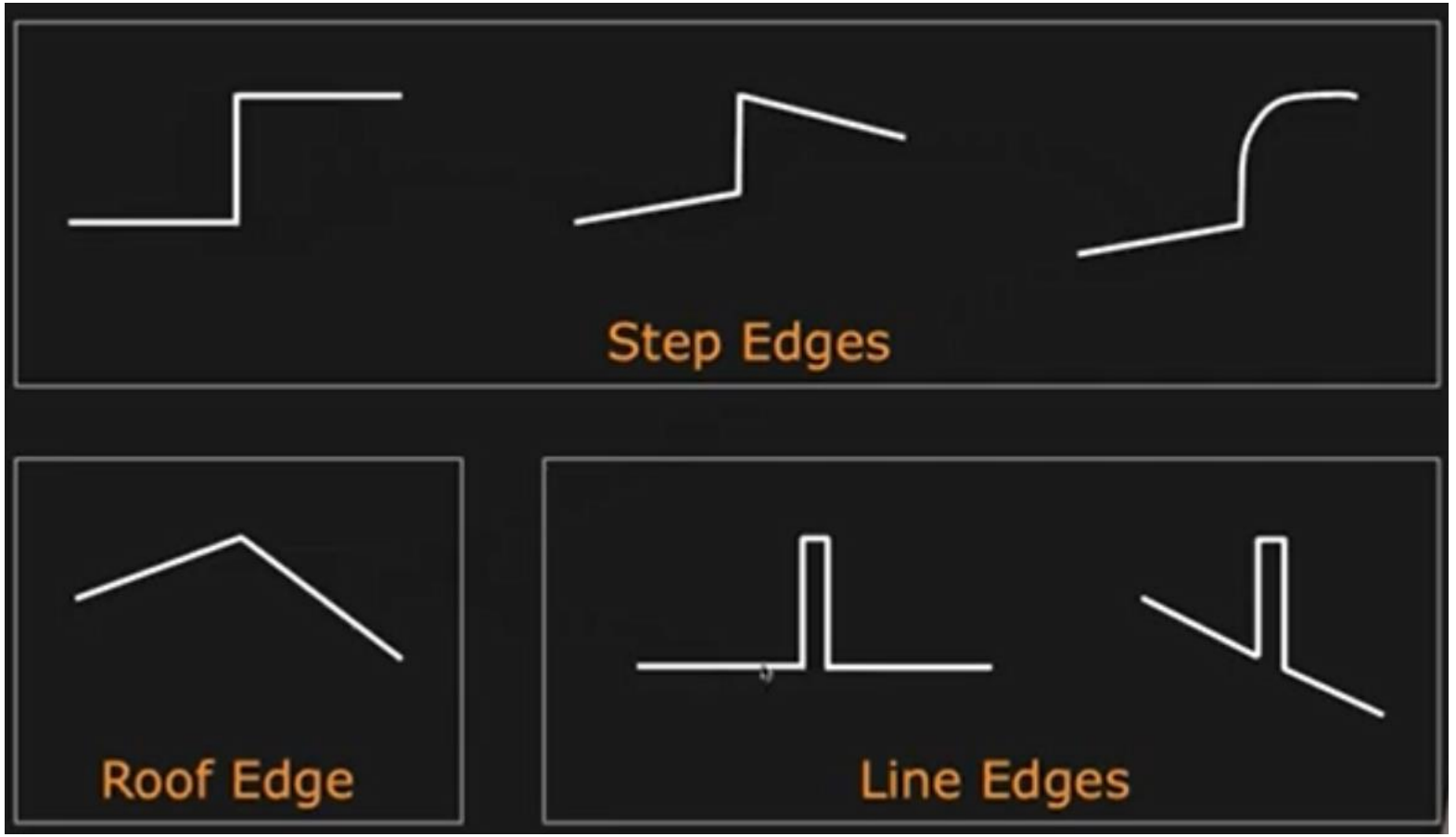


Images as functions…

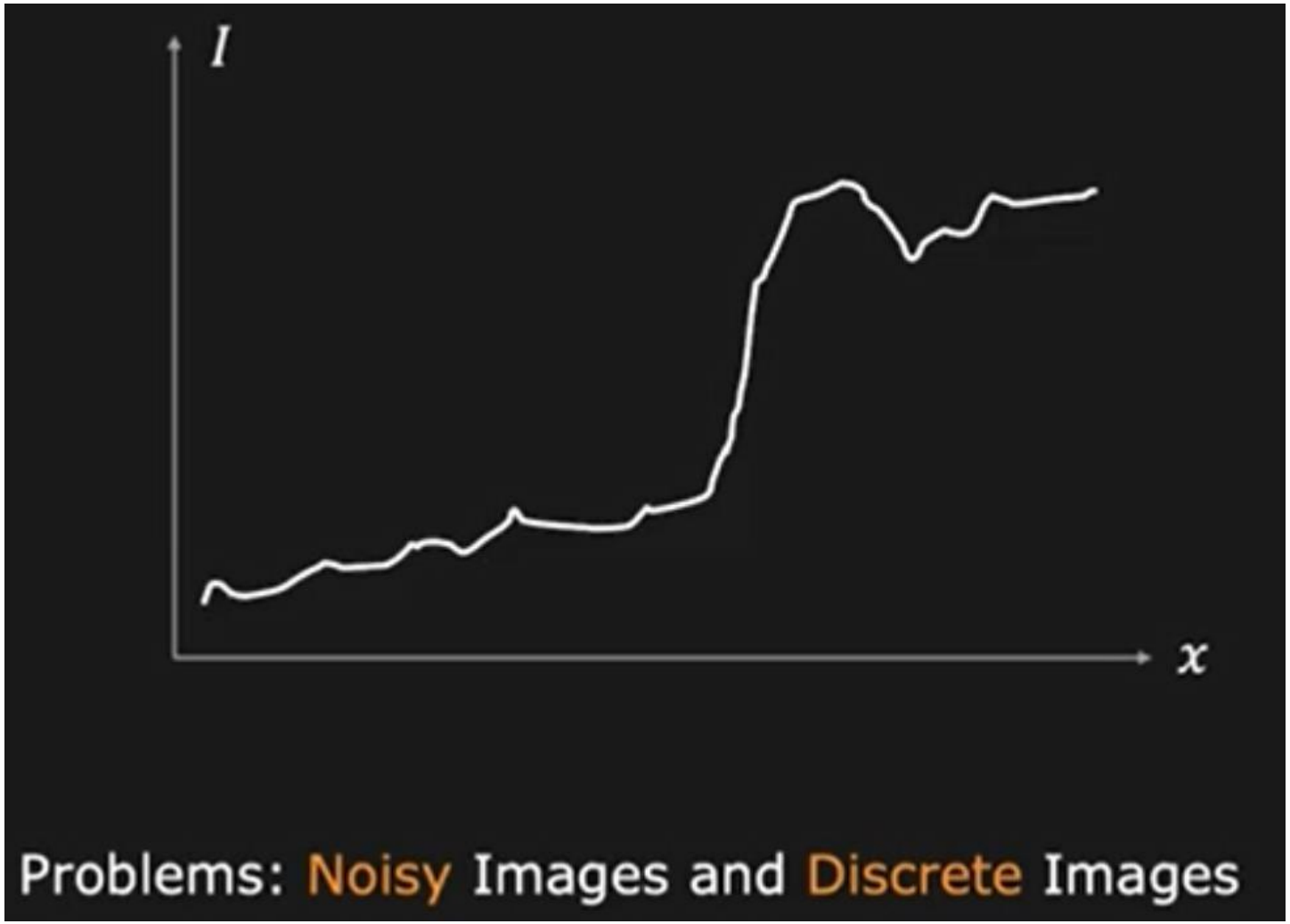


- Edges look like steep cliffs

Type of an Edge



Real Edge



Edge Detector

We want an Edge Operator that produces:

- Edge Position
- Edge Magnitude (Strength)
- Edge Orientation (Direction)

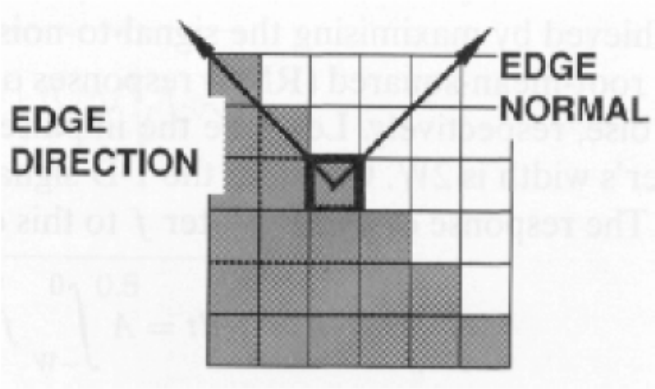
Performance Requirements:

- High Detection Rate
- Good Localization
- Low Noise Sensitivity

Edge Descriptors

Edge descriptors

- Edge normal: unit vector in the direction of maximum intensity change.
- Edge direction: unit vector along edge (perpendicular to edge normal).
- Edge position or center: the image position at which the edge is located.
- Edge strength or magnitude: local image contrast along the normal.



Edge Detection

Convert a 2D image into a set of points where image intensity changes rapidly.

Topics:

(1) What is an Edge?

(2) Edge Detection Using Gradients (Uses First Derivative of Image)

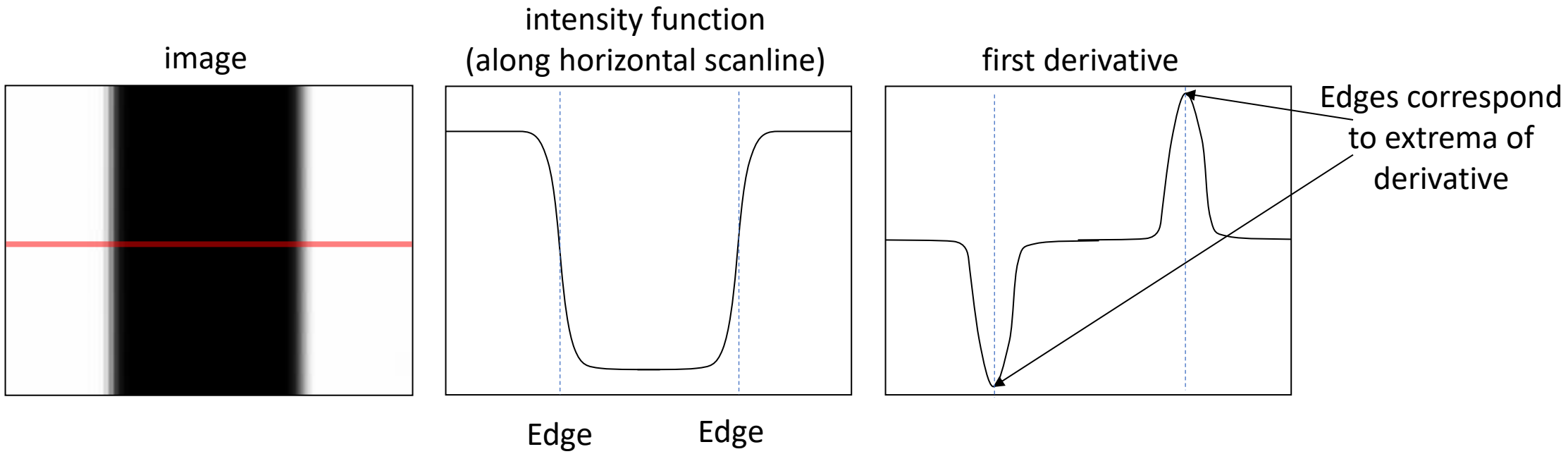
(3) Edge Detection Using Laplacian (Uses Second Derivative of Image)

(4) Canny Edge Detector

(5) Corner Detection

Edge Detection: Characterizing edges using 1st Derivative

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



Source: L. Lazebnik

Basic Calculus: Derivative of a Continuous function represents the amount of change in the function

Edge Detection using 1st Derivative

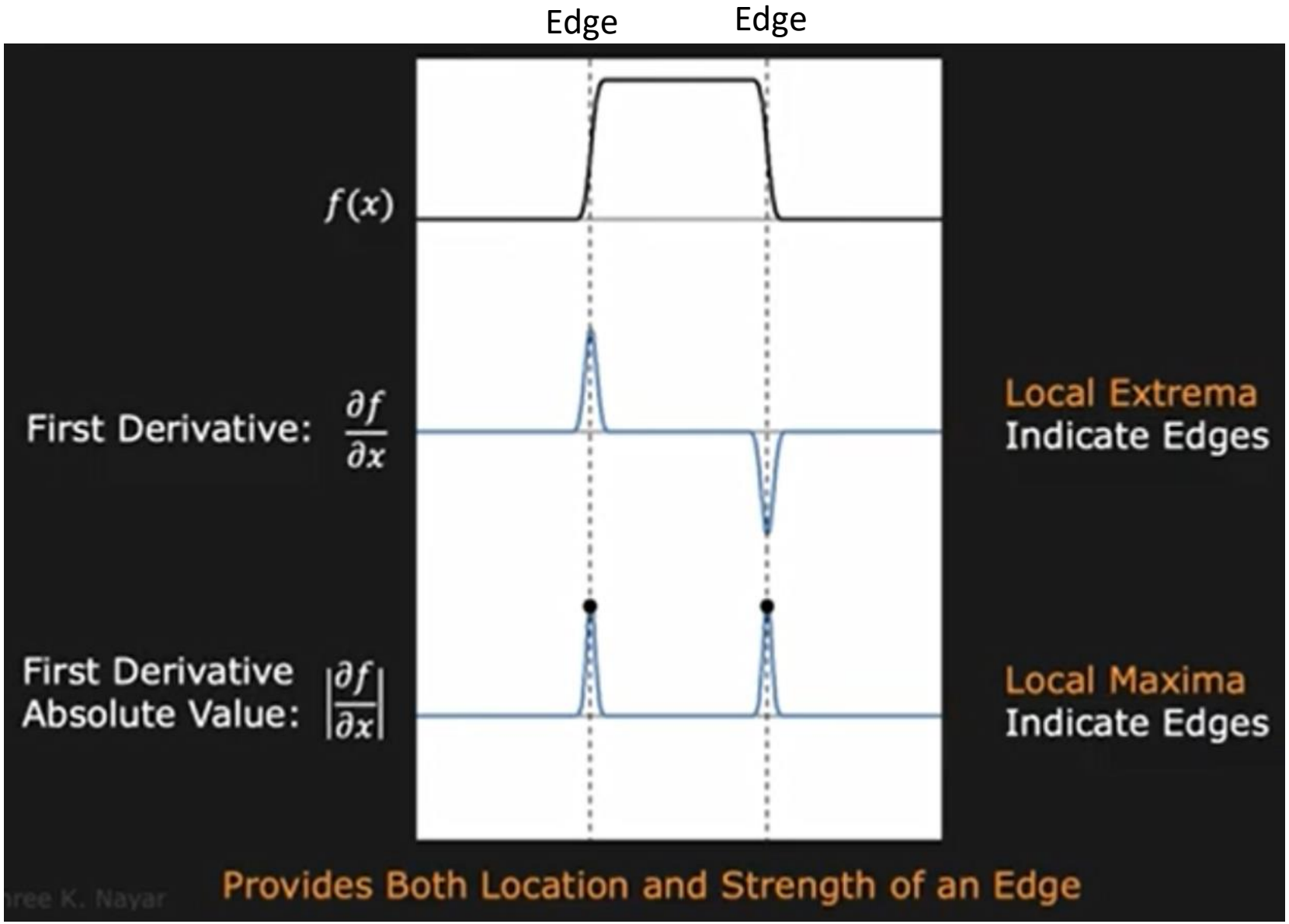


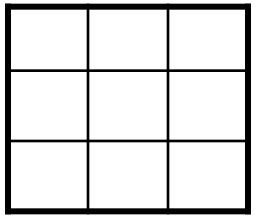
Image derivatives

- How can we differentiate a *digital* image F[x,y]?
  - Option 1: reconstruct a continuous image, *f,* then compute the derivative
  - Option 2: take discrete derivative (finite difference)

$$\frac{\partial f}{\partial x}[x, y] \approx F[x+1, y] - F[x, y]$$

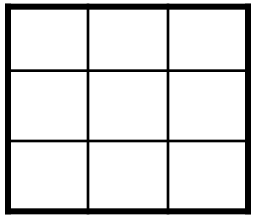
How would you implement this as a linear filter?

$\frac{\partial f}{\partial x}$:



H_x

$\frac{\partial f}{\partial y}$:



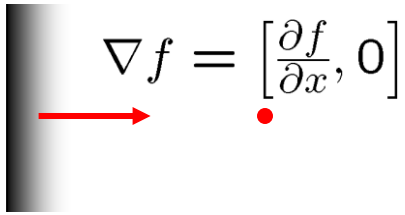
H_y

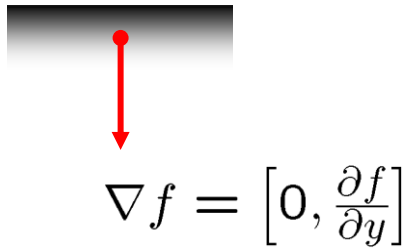
Source: S. Seitz

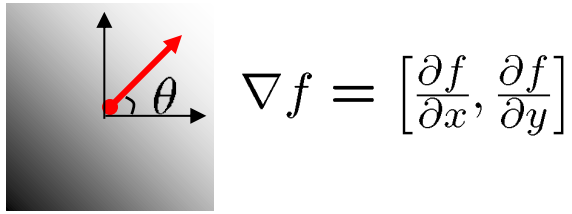
Gradient (∇) of an Image

The *gradient* of an image (del Operator): $\nabla f = \left[\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right]$

The gradient (partial derivatives) points in the direction of most rapid increase in intensity







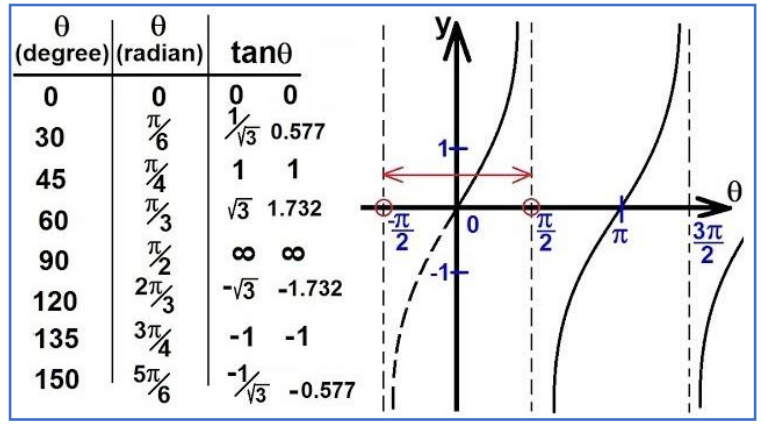
The *edge strength* is given by the gradient magnitude:

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

The gradient direction is given by:

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

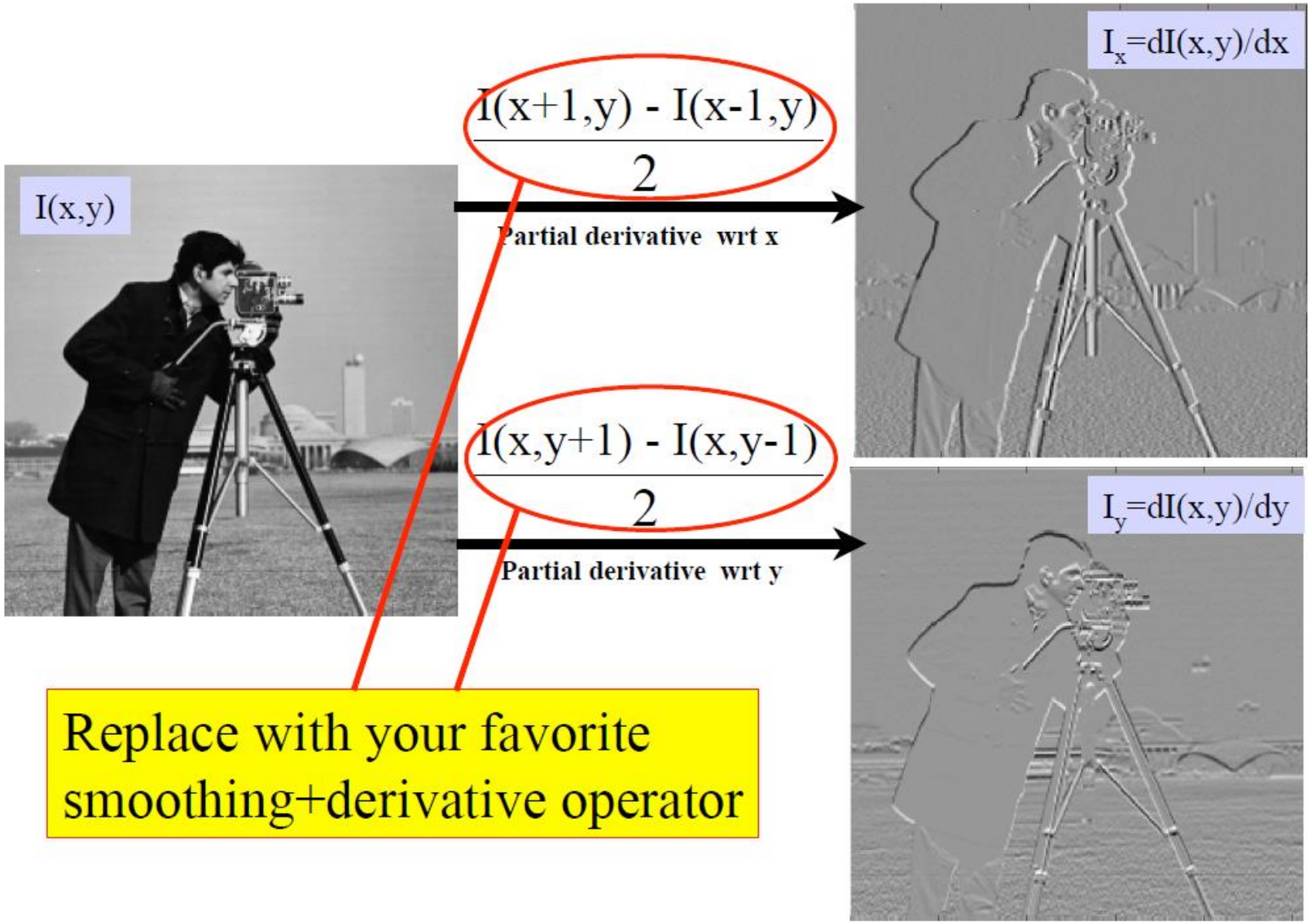
- how does this relate to the direction of the edge?



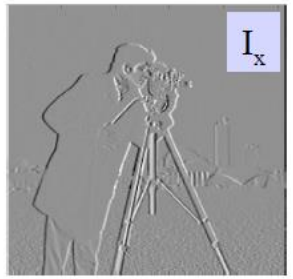
θ (degree)	θ (radian)	tanθ
0	0	0
30	π/6	1/√3 0.577
45	π/4	1 1
60	π/3	√3 1.732
90	π/2	∞ ∞
120	2π/3	-√3 -1.732
135	3π/4	-1 -1
150	5π/6	-1/√3 -0.577

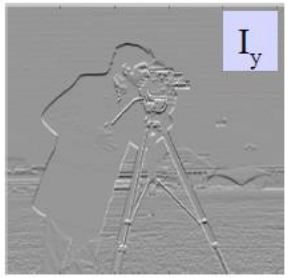
Source: Steve Seitz

Compute Spatial Image Gradients



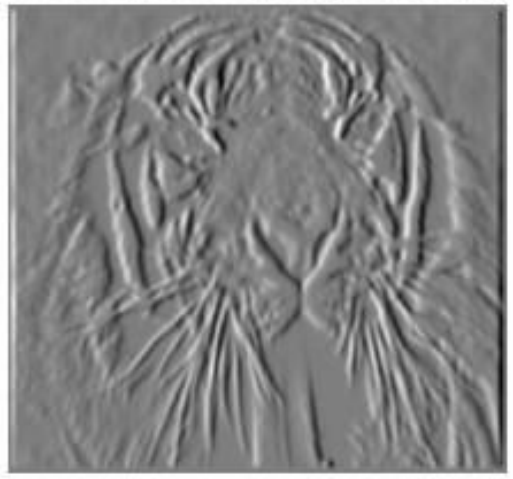
Compute Gradient Magnitude

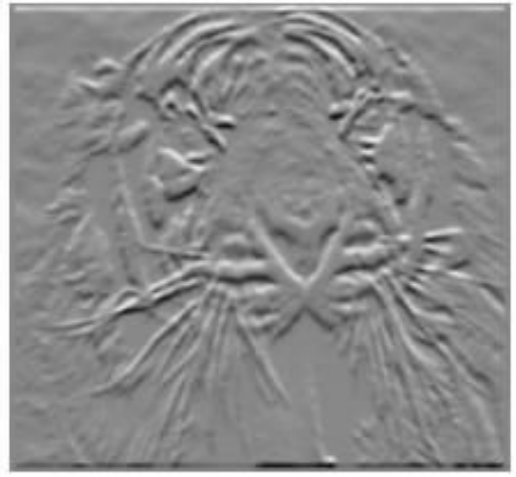




Magnitude of gradient
sqrt(Ix.^2 + Iy.^2)

Measures steepness of slope at each pixel
(= edge contrast)



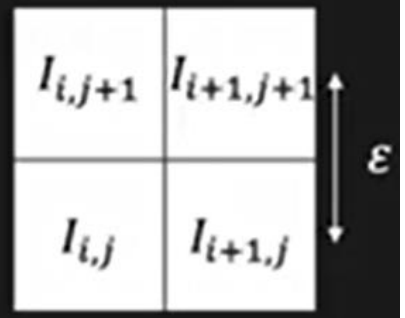


Discrete Gradient (∇) Operator

Finite difference approximations:

$$\frac{\partial I}{\partial x} \approx \frac{1}{2\varepsilon}\left(\left(I_{i+1,j+1} - I_{i,j+1}\right) + \left(I_{i+1,j} - I_{i,j}\right)\right)$$

$$\frac{\partial I}{\partial y} \approx \frac{1}{2\varepsilon}\left(\left(I_{i+1,j+1} - I_{i+1,j}\right) + \left(I_{i,j+1} - I_{i,j}\right)\right)$$



Can be implemented as Convolution!

$$\frac{\partial}{\partial x} \approx \frac{1}{2\varepsilon}\begin{bmatrix} -1 & 1 \\ -1 & 1 \end{bmatrix} \qquad \frac{\partial}{\partial y} \approx \frac{1}{2\varepsilon}\begin{bmatrix} 1 & 1 \\ -1 & -1 \end{bmatrix}$$

Comparing Edge Operators

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

Roberts (2 x 2):

0	1
-1	0

1	0
0	-1

Sobel (3 x 3):

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

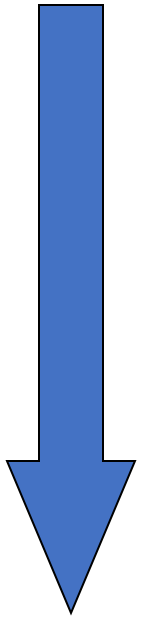
1	1	1
0	0	0
-1	-1	1

Sobel (5 x 5):

-1	-2	0	2	1
-2	-3	0	3	2
-3	-5	0	5	3
-2	-3	0	3	2
-1	-2	0	2	1

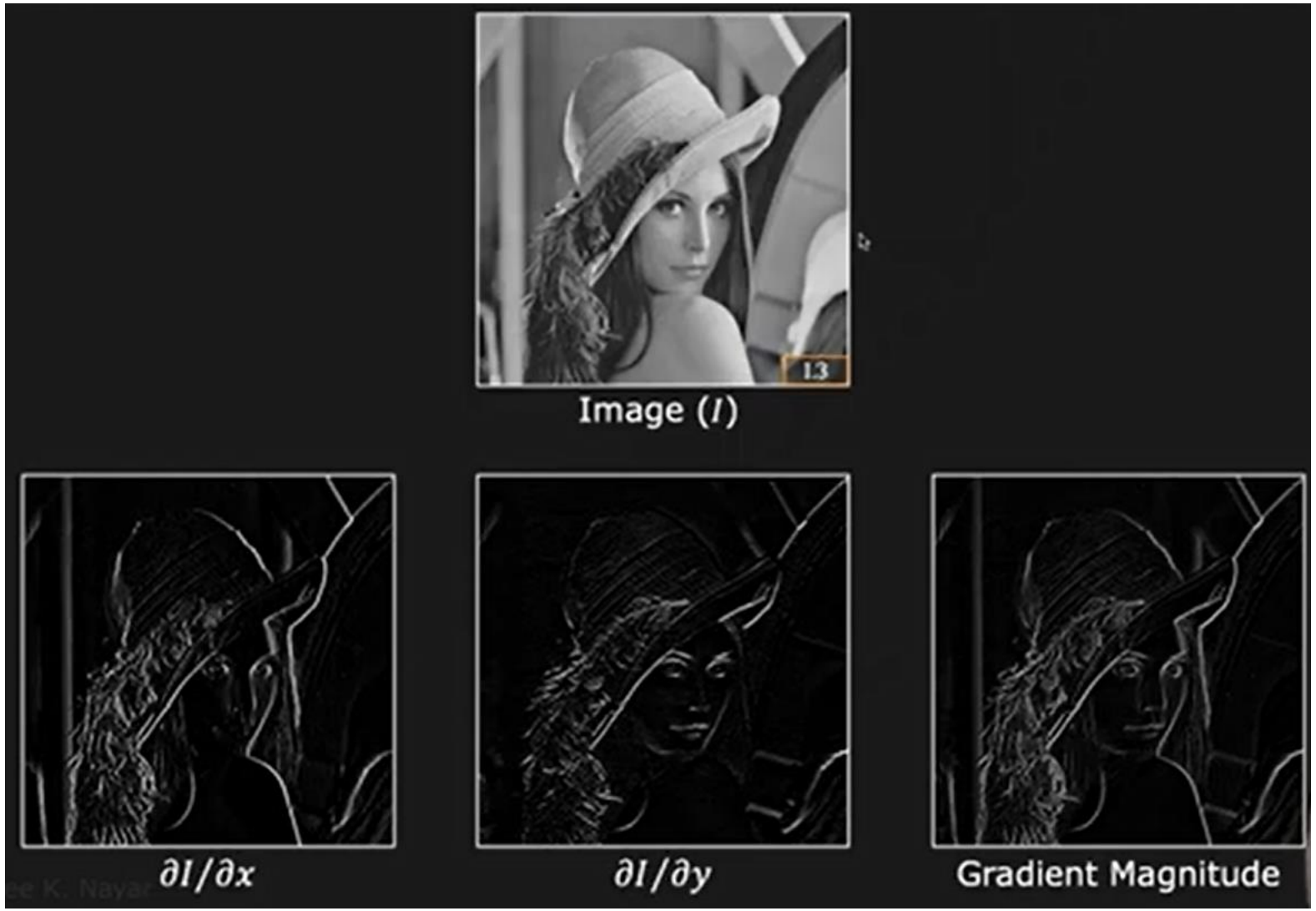
1	2	3	2	1
2	3	5	3	2
0	0	0	0	0
-2	-3	-5	-3	-2
-1	-2	-3	-2	-1

Good Localization
Noise Sensitive
Poor Detection



Poor Localization
Less Noise Sensitive
Good Detection

Gradient (∇) Sobel Filter



Edge Thresholding

Standard: (Single Threshold T)

$\|\nabla I(x,y)\| < T$ Definitely Not an Edge

$\|\nabla I(x,y)\| \geq T$ Definitely an Edge

Hysteresis Based: (Two Thresholds $T_0 < T_1$)

$\|\nabla I(x,y)\| < T_0$ Definitely Not an Edge

$\|\nabla I(x,y)\| \geq T_1$ Definitely an Edge

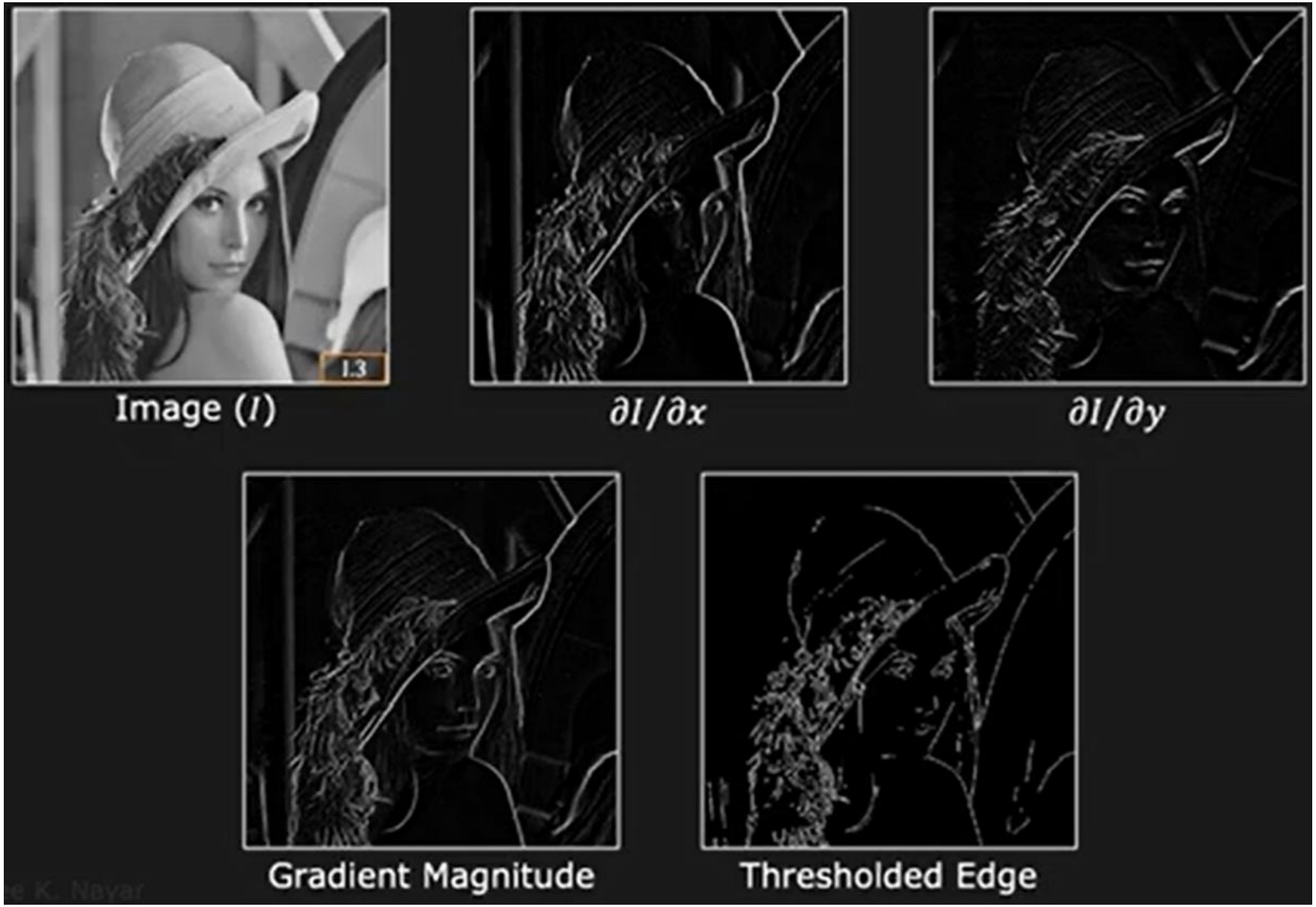
$T_0 \leq \|\nabla I(x,y)\| < T_1$ Is an Edge if a Neighboring Pixel is Definitely an Edge

Simple Edge Detection Using Gradients

A simple edge detector using gradient magnitude

- Compute gradient vector at each pixel by convolving image with horizontal and vertical derivative filters
- Compute gradient magnitude at each pixel
- If magnitude at a pixel exceeds a threshold, report a possible edge point.

Gradient (∇) Sobel Filter with Thresholding



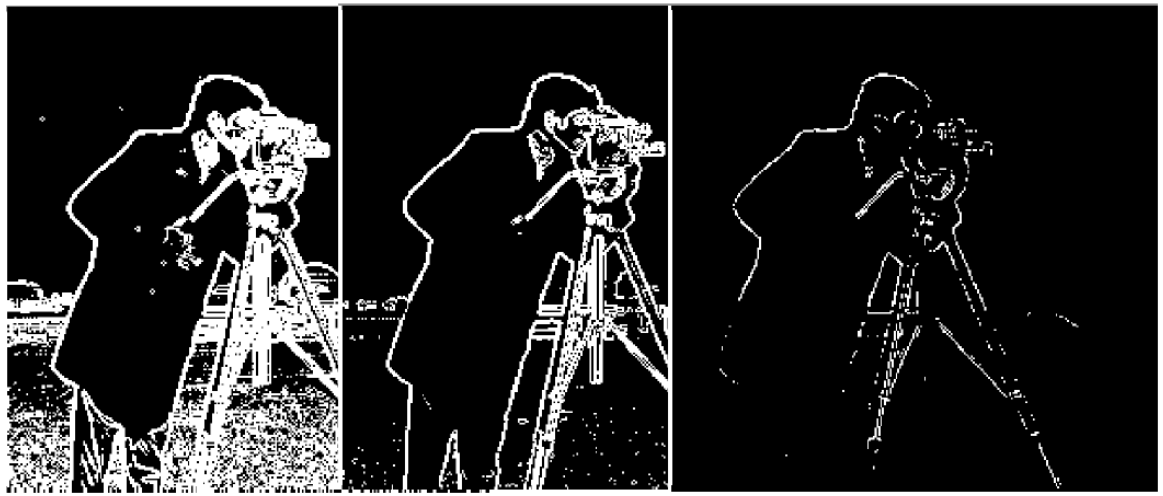
Threshold to find edge pixels

Input Image

Threshold
Mag > 30

Binary Edge Image

How should we choose the threshold?



> 10 > 30 > 80

Issues to address

❖Trade-off: Smoothing vs Localization

There is ALWAYS a tradeoff between smoothing and good edge localization!

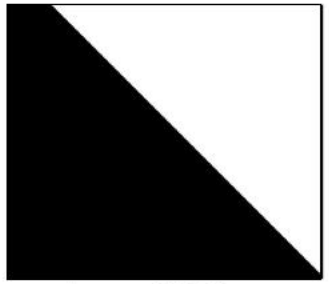
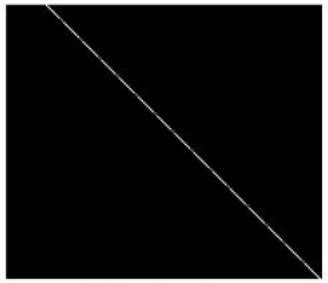


Image with Edge



Edge Location

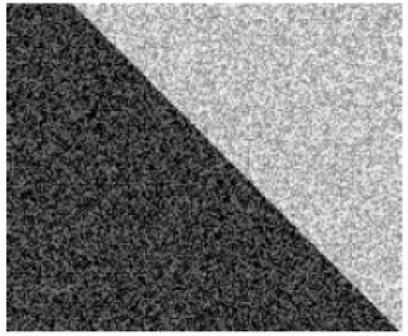
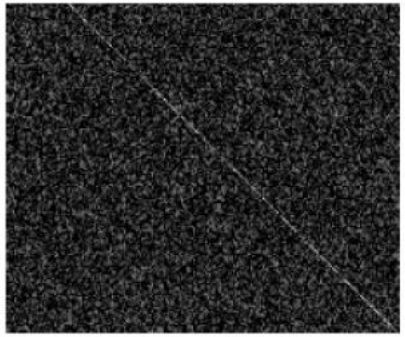


Image + Noise

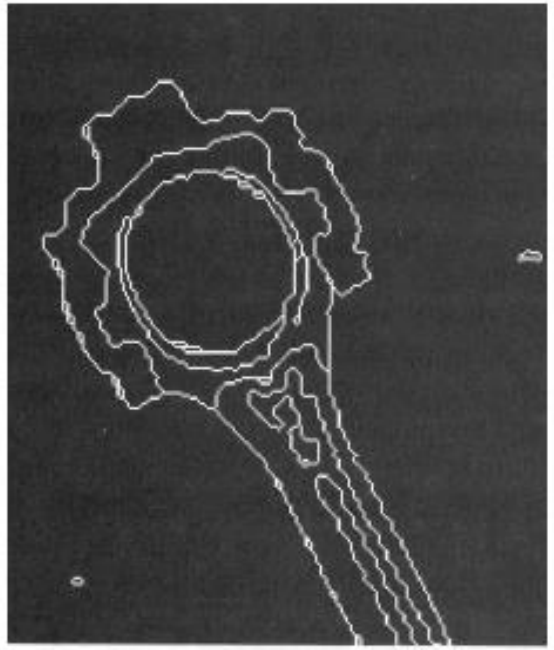


Derivatives detect edge *and* noise

Smoothed derivative removes noise, but blurs edge

❖Edge Linking and Thinning

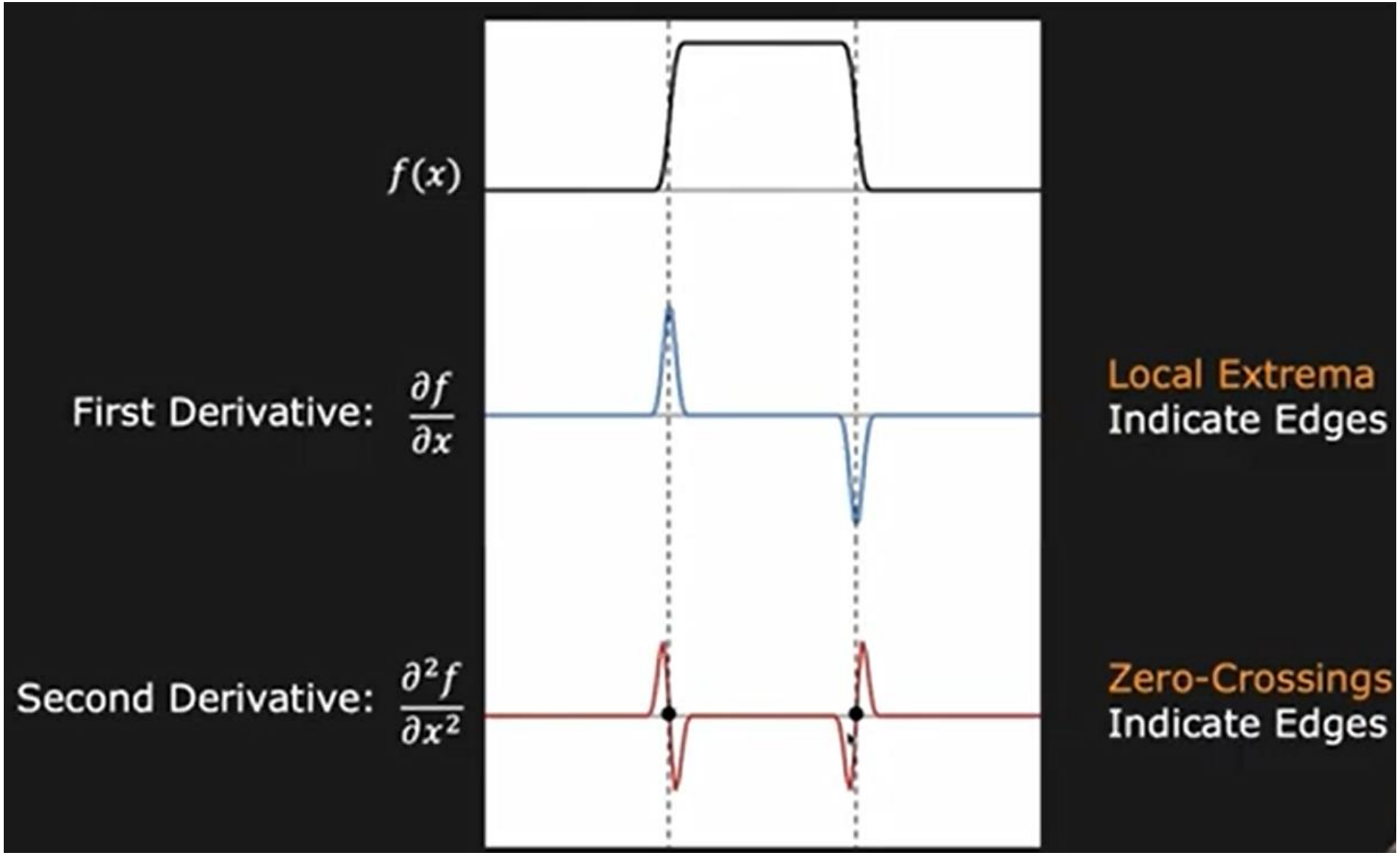
Smoothing + thresholding gives us a binary mask with “thick” edges



we want thin, one-pixel wide, connected contours

Edge Detection using Laplacian

Edge Detection using 2nd Derivative



Laplacian (∇^2) Operator

Laplacian: Sum of Pure Second Derivatives

$$\nabla^2 I = \frac{\partial^2 I}{\partial x^2} + \frac{\partial^2 I}{\partial y^2}$$

Pronounced as "Del Square I"

- Edges are "zero-crossings" in Laplacian of image
- Laplacian does not provide directions of edges

Discrete Laplacian (∇^2) Operator

Finite difference approximations:

$$\frac{\partial^2 I}{\partial x^2} \approx \frac{1}{\varepsilon^2}\left(I_{i-1,j} - 2I_{i,j} + I_{i+1,j}\right)$$

$$\frac{\partial^2 I}{\partial y^2} \approx \frac{1}{\varepsilon^2}\left(I_{i,j-1} - 2I_{i,j} + I_{i,j+1}\right)$$

$$\nabla^2 I = \frac{\partial^2 I}{\partial x^2} + \frac{\partial^2 I}{\partial y^2}$$

$I_{i-1,j+1}$	$I_{i,j+1}$	$I_{i+1,j+1}$
$I_{i-1,j}$	$I_{i,j}$	$I_{i+1,j}$
$I_{i-1,j-1}$	$I_{i,j-1}$	$I_{i+1,j-1}$

ε

Convolution Mask:

$$\nabla^2 \approx \frac{1}{\varepsilon^2}\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

OR

$$\nabla^2 \approx \frac{1}{6\varepsilon^2}\begin{bmatrix} 1 & 4 & 1 \\ 4 & -20 & 4 \\ 1 & 4 & 1 \end{bmatrix}$$

(More Accurate,

ee K. Nayar

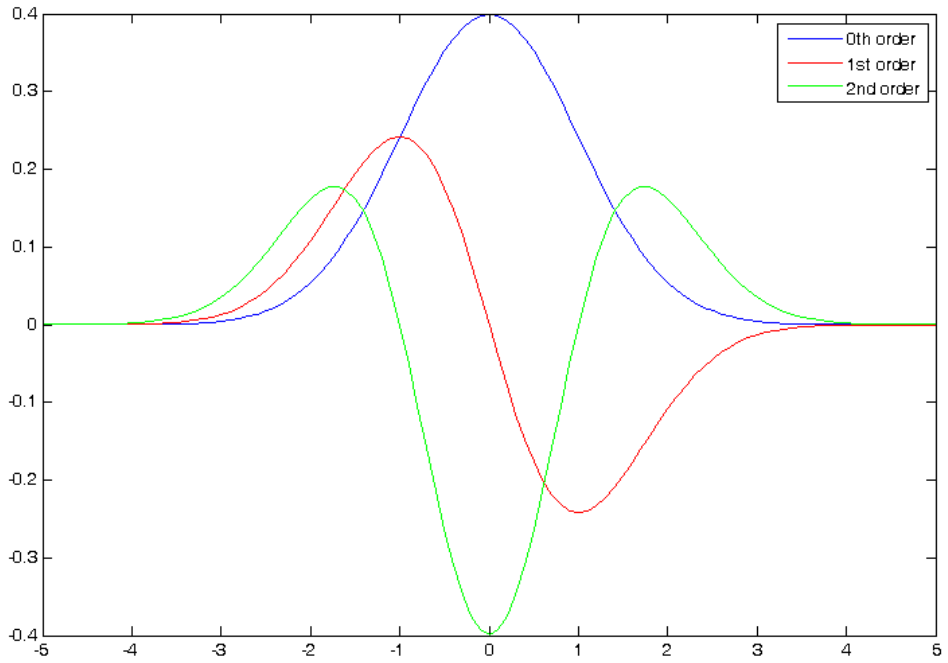
Laplacian Edge Detector

Derivative of Gaussian and Laplacian of Gaussian

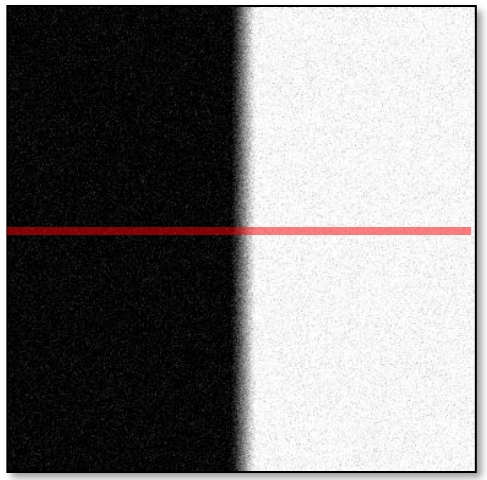
The 1D Gaussian and its derivatives

$$G_\sigma(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{x^2}{2\sigma^2}}$$

$$G'_\sigma(x) = \frac{d}{dx} G_\sigma(x) = -\frac{1}{\sigma}\left(\frac{x}{\sigma}\right) G_\sigma(x)$$

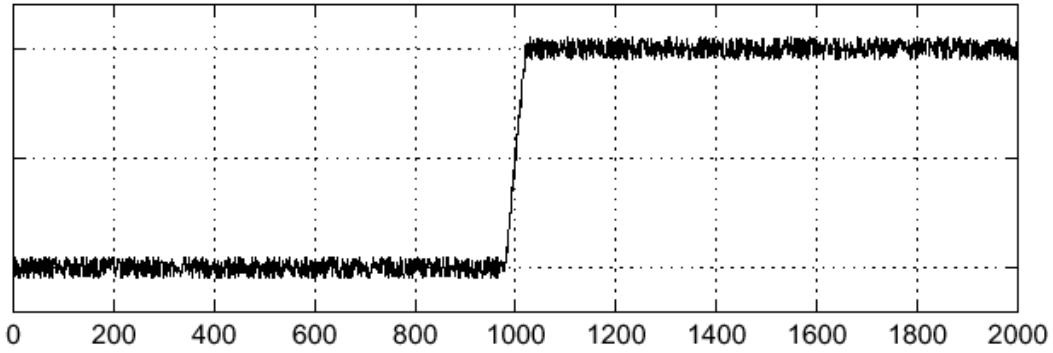


Effects of noise

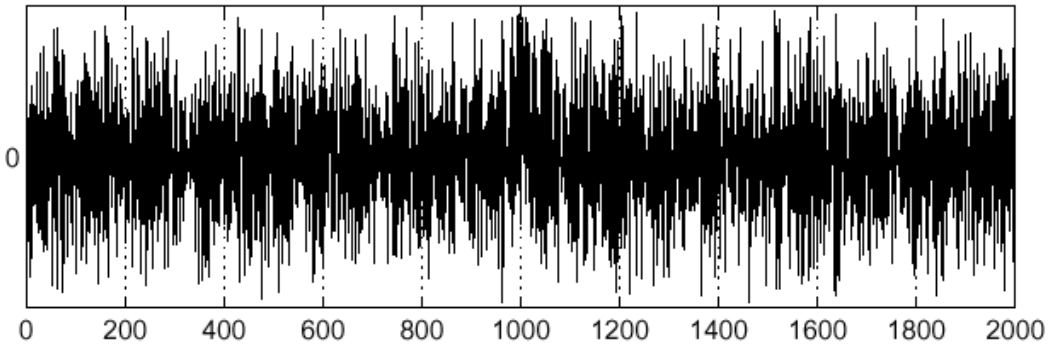


Noisy input image

$f(x)$



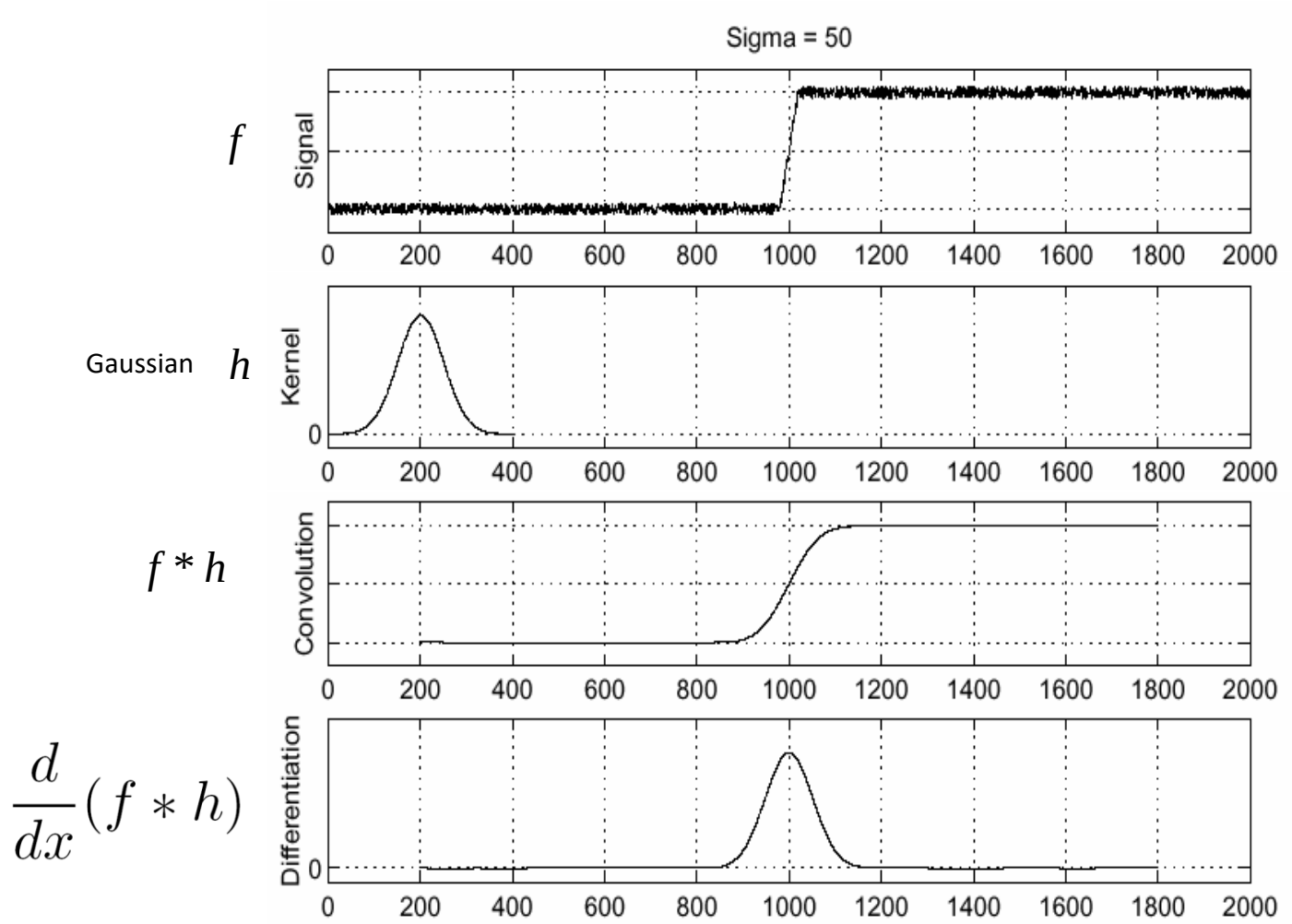
$\frac{d}{dx}f(x)$



Where is the edge?

Source: S. Seitz

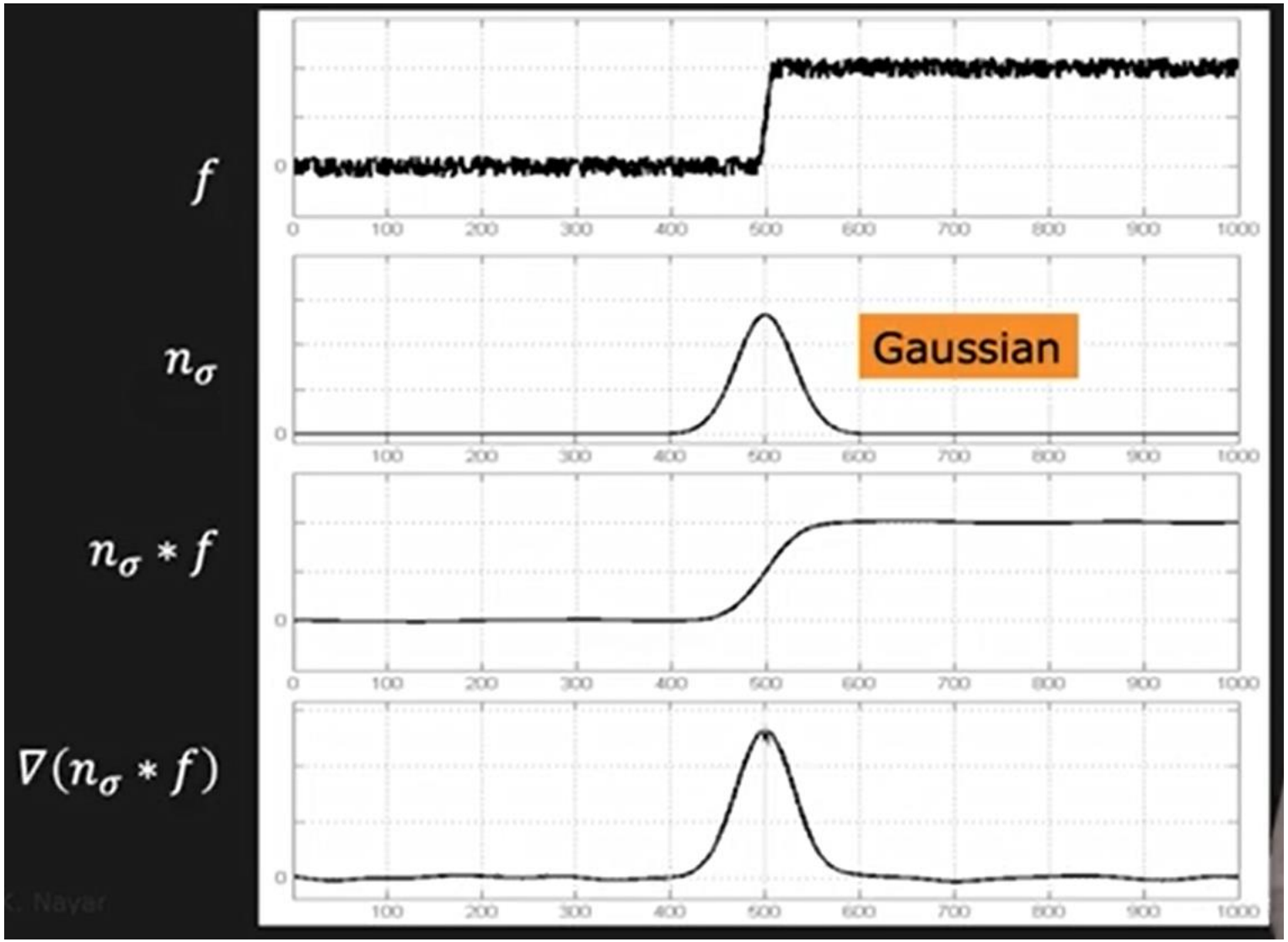
Solution: smooth first



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

Source: S. Seitz

Solution: Gaussian Smooth First



Derivative of Gaussian: (∇ (n_σ))

- Differentiation is convolution, and convolution is associative:

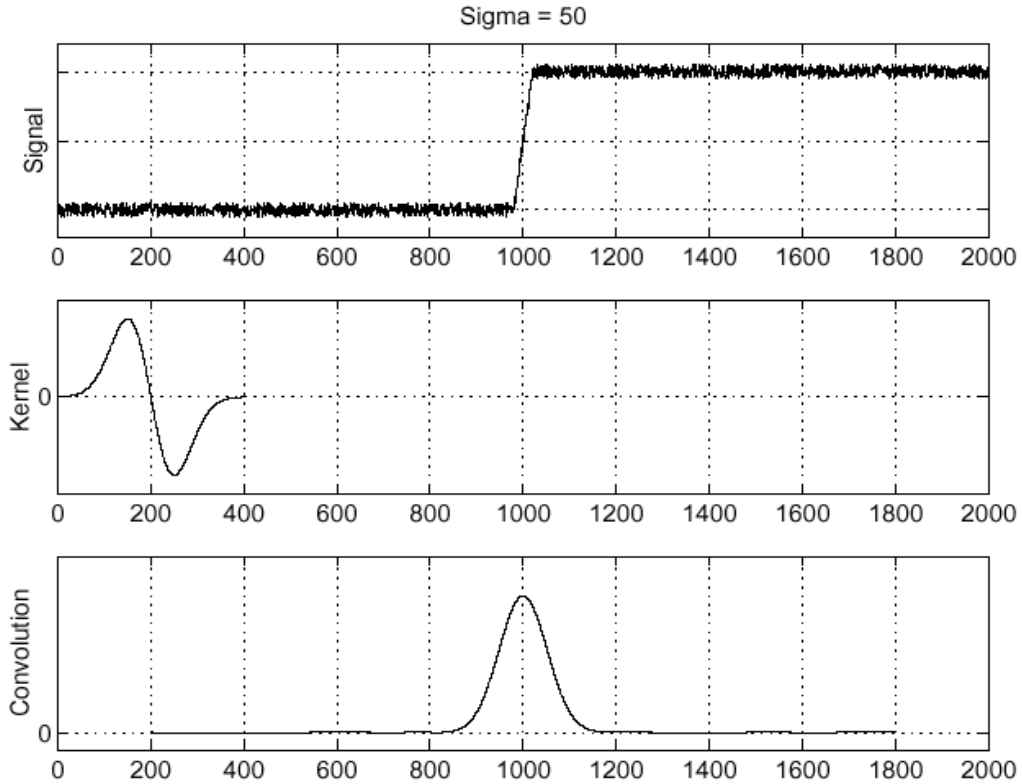
$$\frac{d}{dx}(f * h) = f * \frac{d}{dx}h$$

- This saves us one operation:

f

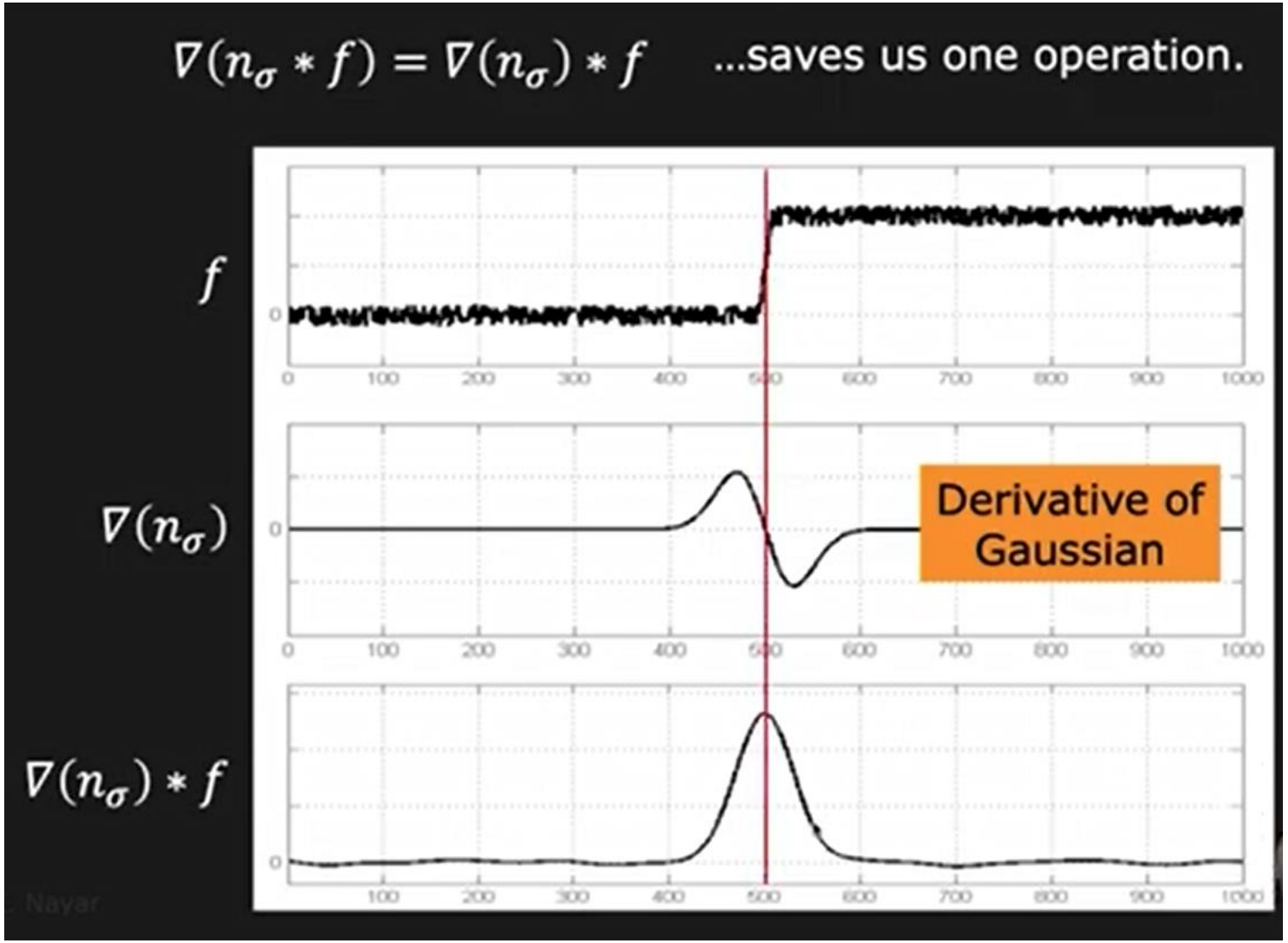
Derivative of Gaussian $\frac{d}{dx}h$

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

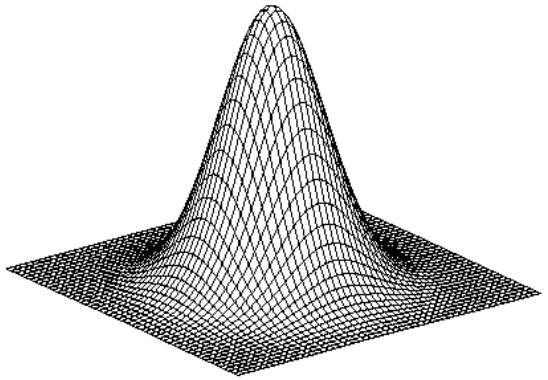


Source: S. Seitz

Derivative of Gaussian ($\nabla (n_\sigma)$)

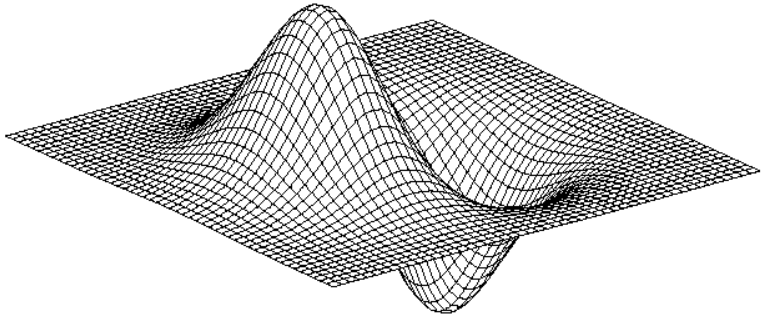


2D edge detection filters



Gaussian

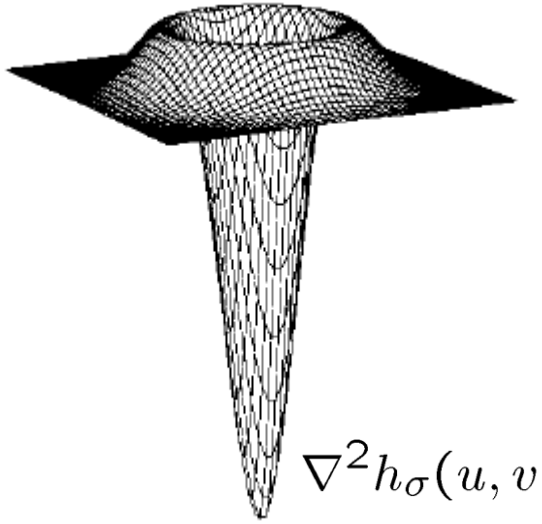
$$h_\sigma(u,v) = \frac{1}{2\pi\sigma^2} e^{-\frac{u^2+v^2}{2\sigma^2}}$$



Derivative of Gaussian (x)

$$\frac{\partial}{\partial x} h_\sigma(u,v)$$

Laplacian of Gaussian

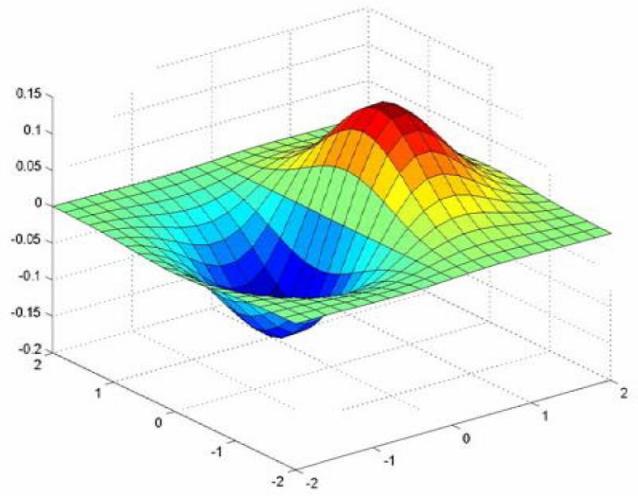


$$\nabla^2 h_\sigma(u,v)$$

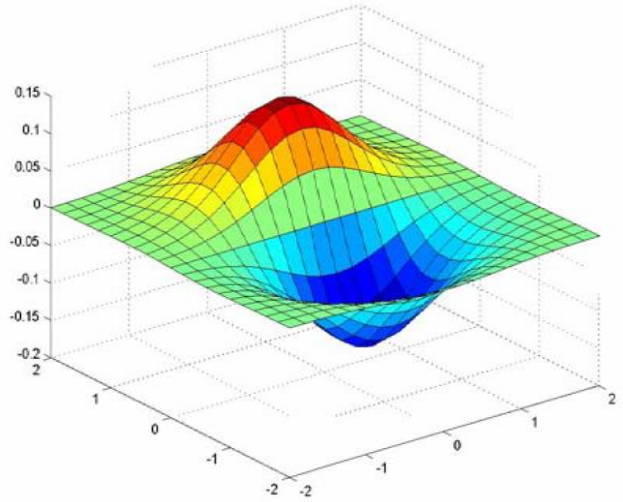
$$\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$$

∇^2 is the **Laplacian** operator

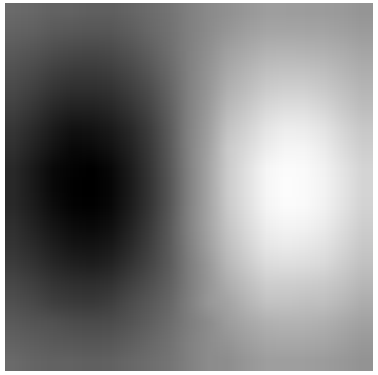
Derivative of Gaussian filter

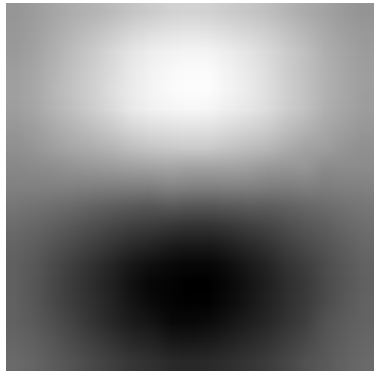


x-direction

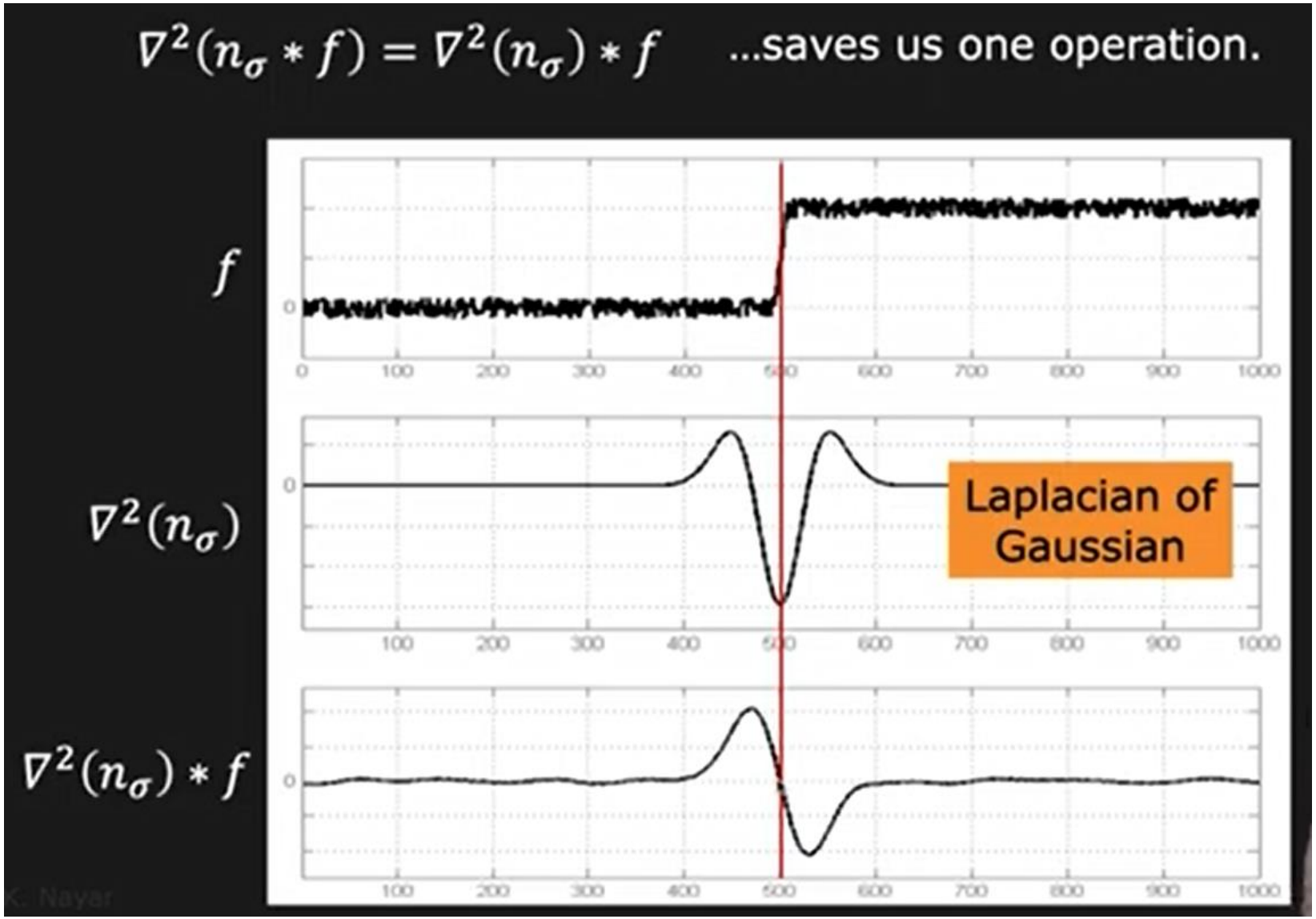


y-direction

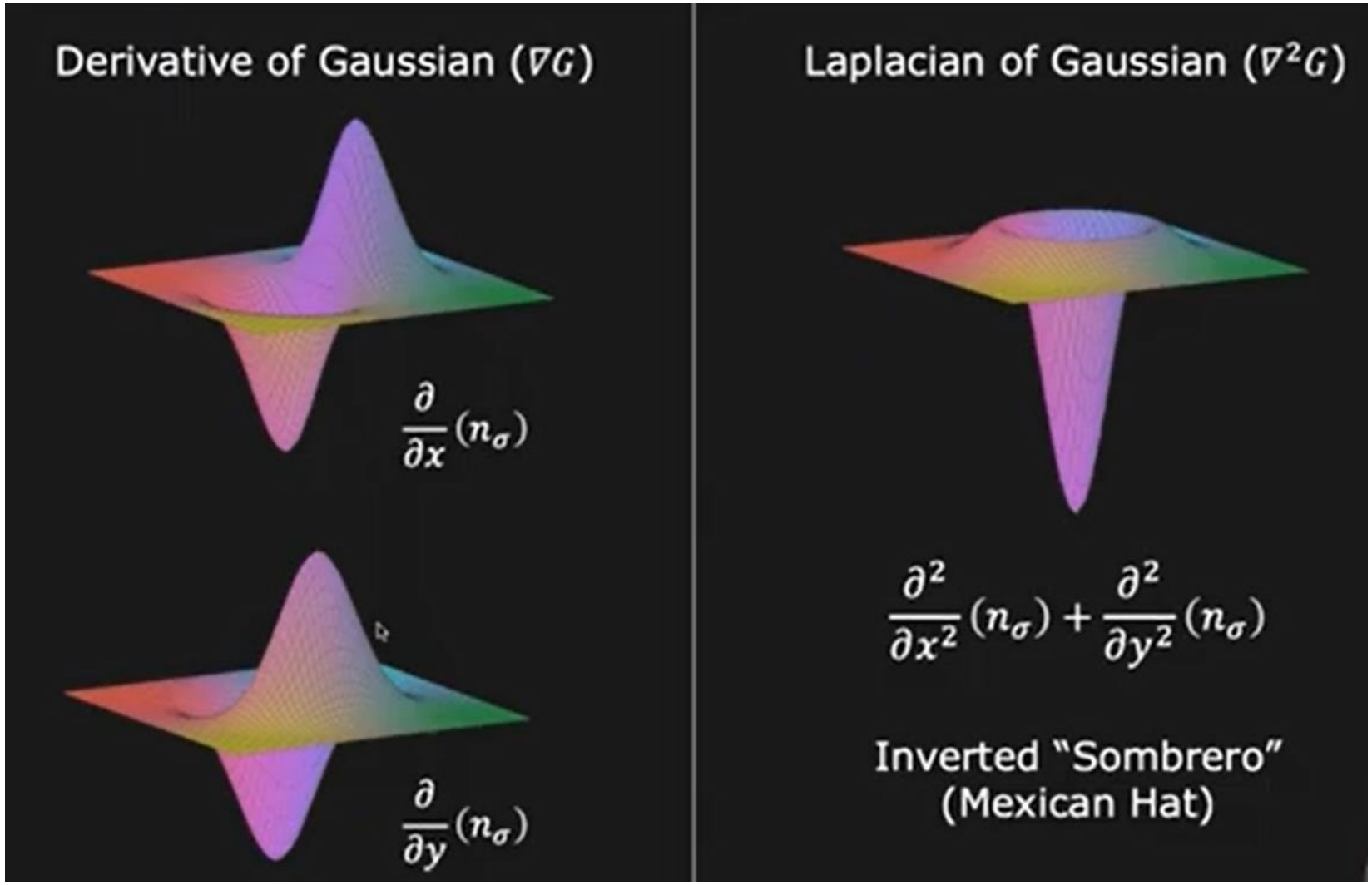




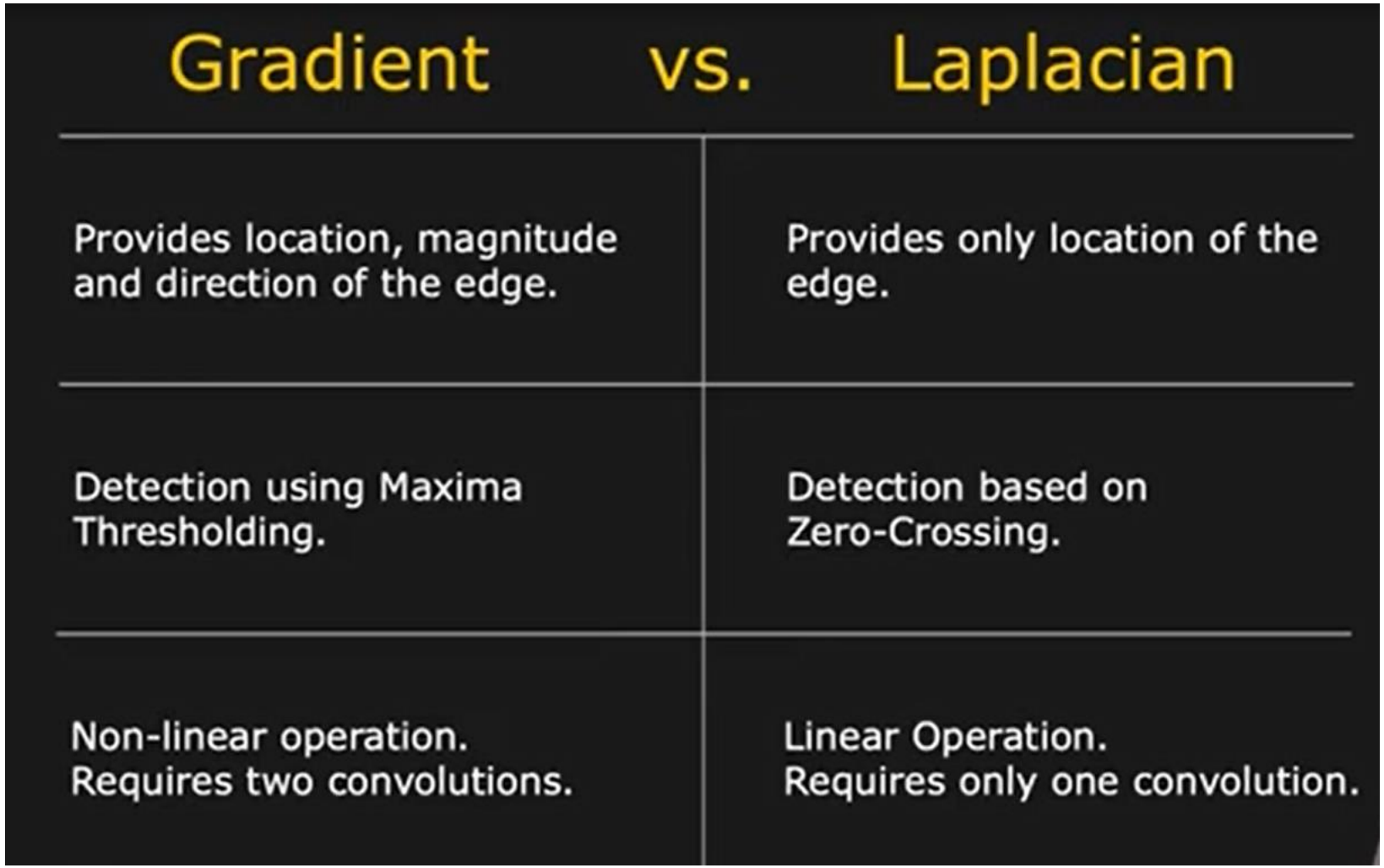
Laplacian of Gaussian (LoG): $\nabla^2 G$ or $\nabla^2 n_\sigma$



Derivative of Gaussian vs Laplacian of Gaussian



Gradient vs Laplacian



Canny Edge Detector

Design of an Optimal Edge Detector

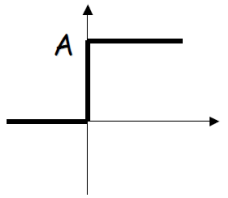
❖Edge detection involves 3 steps:

- ➢ Noise smoothing
- ➢ Edge enhancement
- ➢ Edge localization

J. Canny formalized these steps to design an *optimal* edge detector

Edge Model (1d)

- An ideal edge can be modeled as an step



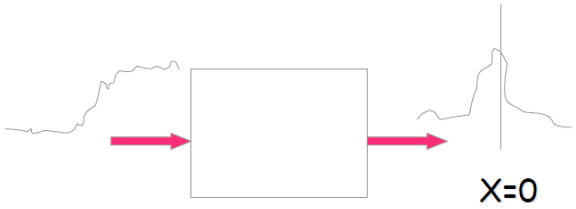
$$G(x) = \begin{cases} 0 & \text{if } x < 0 \\ A & \text{if } x \le 0 \end{cases}$$

- Additive, White Gaussian Noise
  - RMS noise amplitude/unit length n_o^2

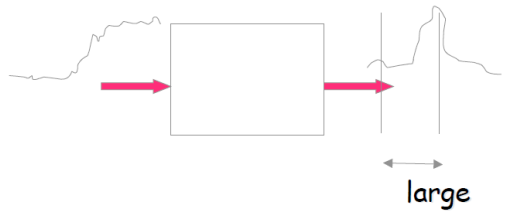
❖Edge detection Performance Criteria:

- Good detection
  - The filter must have a stronger response at the edge location (x=0) than to noise

- Good Localization
  - The filter response must be maximum very close to x=0

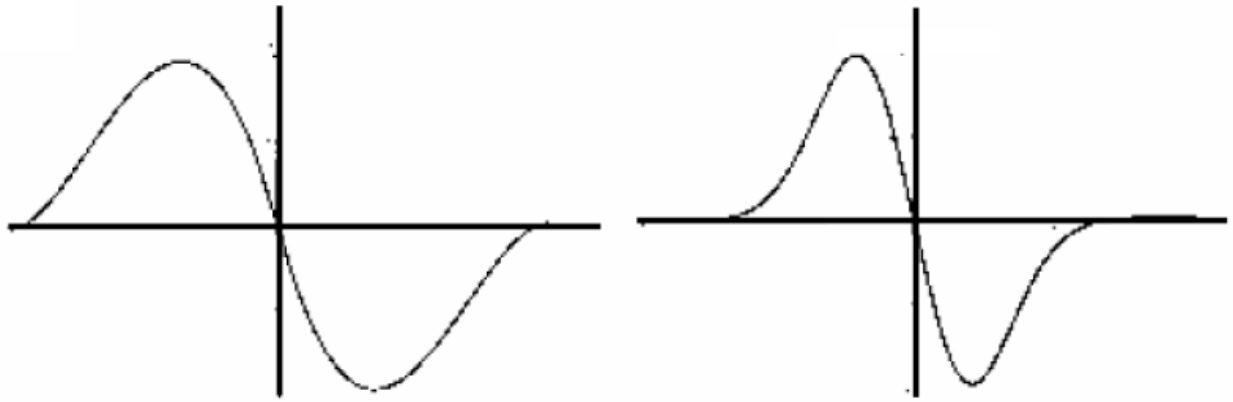


- Low False Positives
  - There should be only one maximum in a reasonable neighborhood of x=0



Canny Edge Detector

- Canny found a linear, continuous filter that maximized the three given criteria.
- There is no closed-form solution for the optimal filter.
- However, it looks VERY SIMILAR to the derivative of a Gaussian.

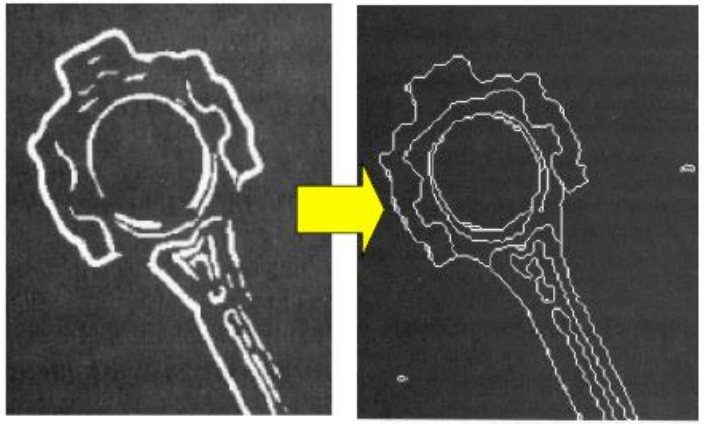


Canny

Derivative of Gaussian

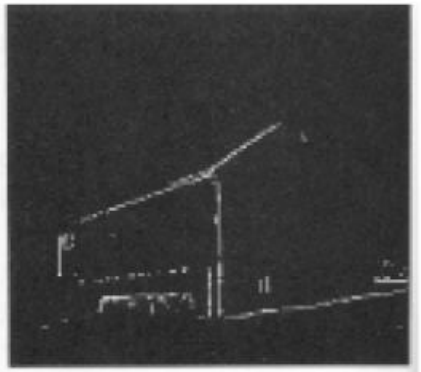
Recall: Practical Issues for Edge Detection

Thinning and linking



Choosing a magnitude threshold

OR



Canny has good answers to all!

The Sobel operator

- Common approximation of Derivative of Gaussian

$\frac{1}{8}$

-1	0	1
-2	0	2
-1	0	1

s_x

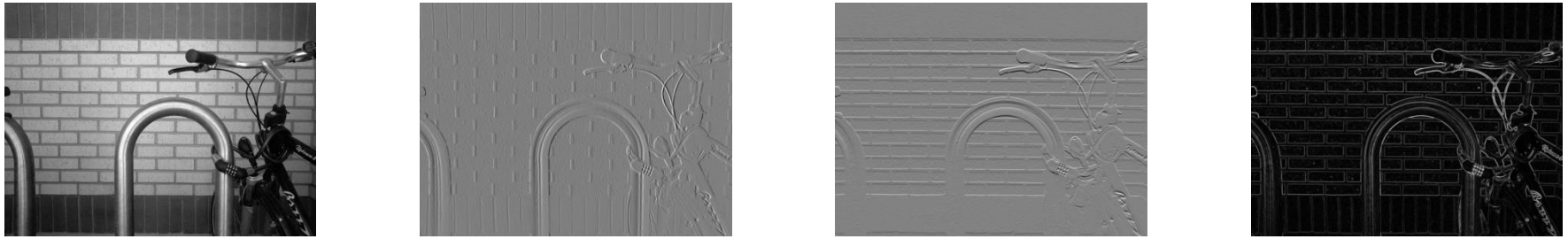
$\frac{1}{8}$

1	2	1
0	0	0
-1	-2	-1

s_y

- The standard definition of the Sobel operator omits the 1/8 term
  - doesn't make a difference for edge detection
  - the 1/8 term **is** needed to get the right gradient magnitude

Sobel operator: example



Source: Wikipedia

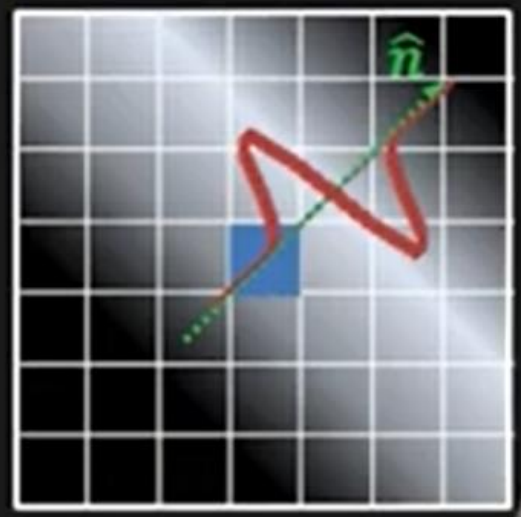
Canny Edge Detector

- Smooth Image with 2D Gaussian: $n_\sigma * I$
- Compute Image Gradient using Sobel Operator: $\nabla n_\sigma * I$
- Find Gradient Magnitude at each pixel: $\|\nabla n_\sigma * I\|$
- Find Gradient Orientation at each Pixel:

$$\hat{n} = \frac{\nabla n_\sigma * I}{\|\nabla n_\sigma * I\|}$$

- Compute Laplacian along the Gradient Direction $\hat{n}$ at each pixel

$$\frac{\partial^2 (n_\sigma * I)}{\partial \hat{n}^2}$$

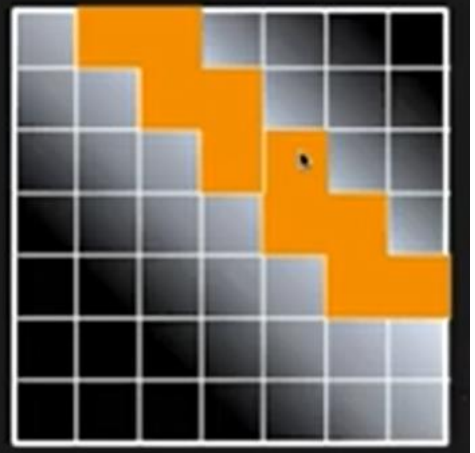


Note: Apply 1-dimensional Laplacian along the direction of the edge (along vector n̂)

Canny Edge Detector

- Smooth Image with 2D Gaussian: $n_\sigma * I$
- Compute Image Gradient using Sobel Operator: $\nabla n_\sigma * I$
- Find Gradient Magnitude at each pixel: $\|\nabla n_\sigma * I\|$
- Find Gradient Orientation at each Pixel:

$$\hat{n} = \frac{\nabla n_\sigma * I}{\|\nabla n_\sigma * I\|}$$



$|\nabla n_\sigma * I|$

- Compute Laplacian along the Gradient Direction $\hat{n}$ at each pixel

$$\frac{\partial^2 (n_\sigma * I)}{\partial \hat{n}^2}$$

- Find Zero Crossings in Laplacian to find the edge location

Demo: https://bigwww.epfl.ch/demo/ip/demos/edgeDetector/

Edge Detection (Scale Space Support)

Edges at different scales of resolution

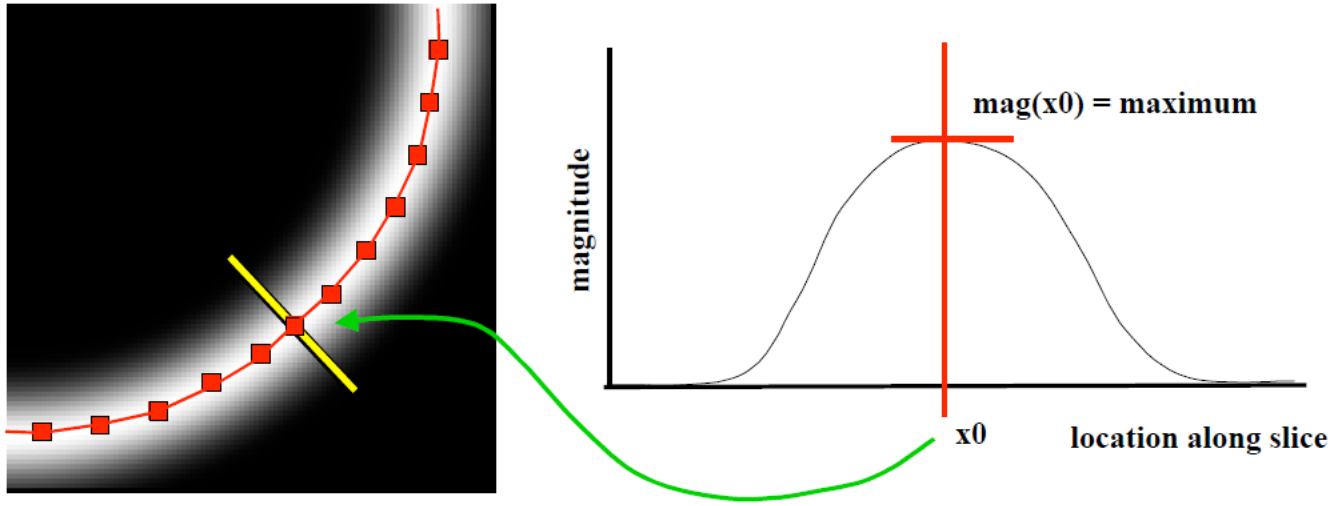
By changing σ, Canny Edge Detector pull out edges that are of interest (at different levels of resolution)

The choice of **σ** depends on desired behavior

- large detects large scale edges
- small detects fine features

Thinning of Edges

❖Do Thinning before thresholding

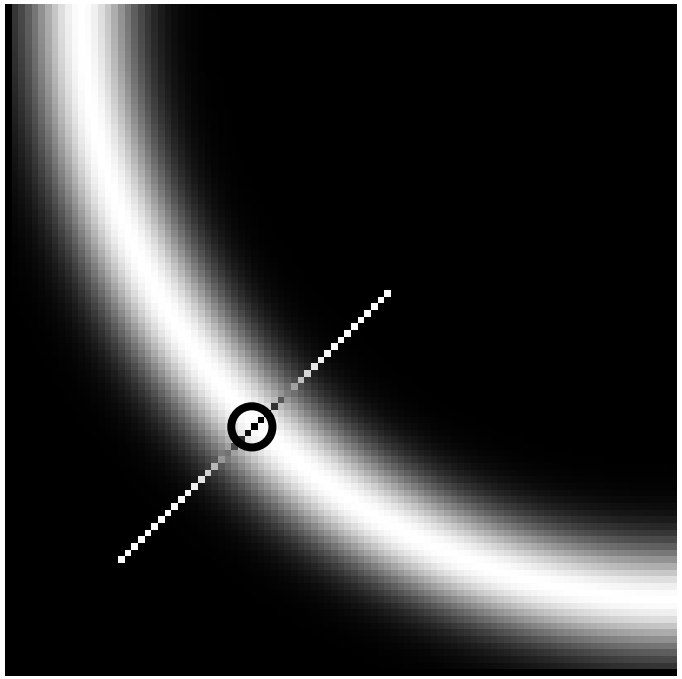


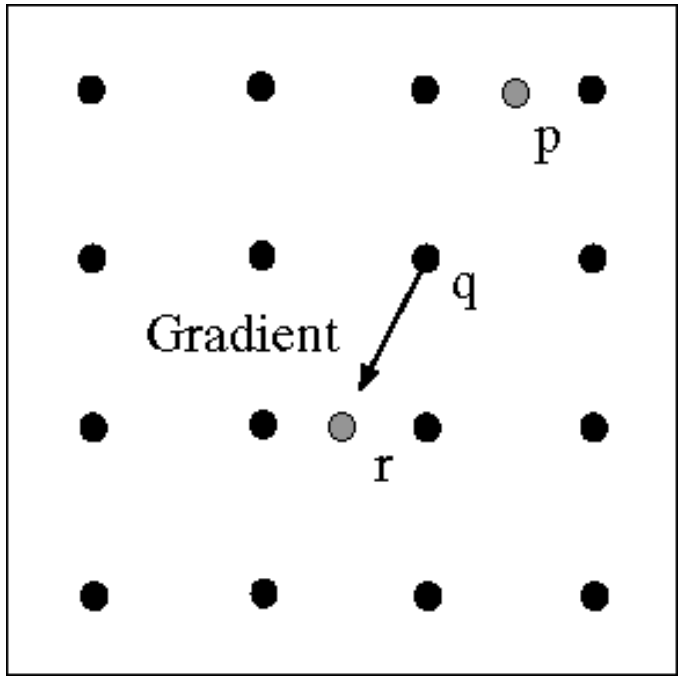
We want to mark points along curve where the magnitude is largest.

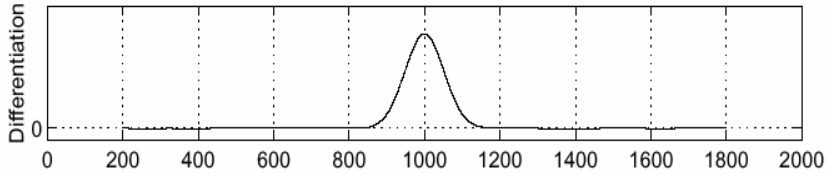
We can do this by looking for a maximum along a 1D intensity slice normal to the curve (non-maximum supression).

These points should form a one-pixel wide curve.

Thin Edges using Non-maximum suppression





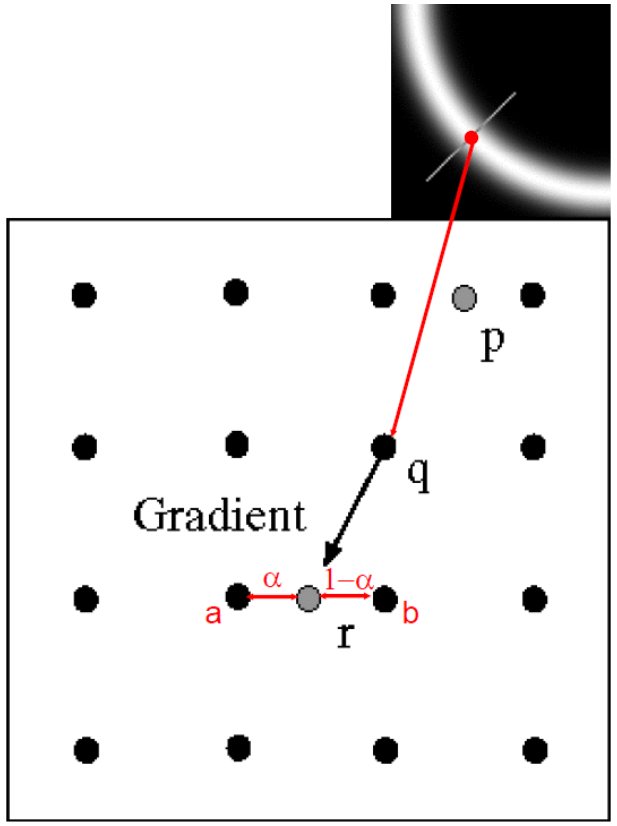


Key Idea: Check if pixel is local maximum along gradient direction

- requires *interpolating* pixels p and r

Thin Edges using Non-maximum suppression

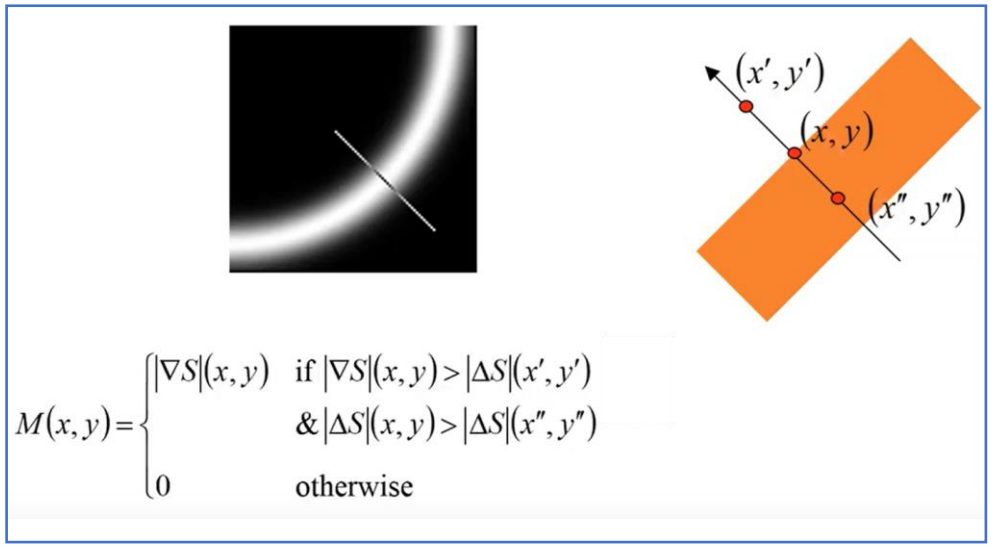
The image magnitude may produce thick edges. Ideally, the final image should have thin edges. Thus, we must perform non maximum suppression to thin out the edges.



Key Idea: In a local neighborhood, for each pixel in input image, whether the magnitude of this edge > neighboring pixels magnitude (see if it is a peak).
Need to consider only neighboring pixels on edge orientation.

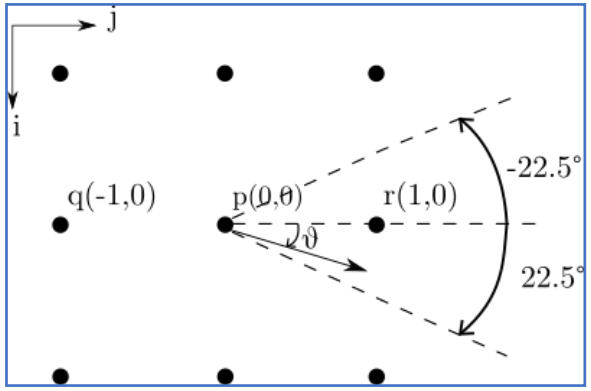
- Non maximum suppression works by finding the pixel with the maximum value in an edge.
- In this image, it occurs when pixel q has an intensity that is larger than both p and r where pixels p and r are the pixels in the gradient direction of q.
- If this condition is true, then we keep the pixel, else we set the pixel to zero (make it a black pixel).
- Non maximum suppression can be achieved by interpolating the pixels for greater accuracy:

$r=\alpha b+(1-\alpha)a$



Non-maximum Suppression with / without Interpolation

Non maximum suppression without interpolation requires us to divide the 3x3 grid of pixels into 8 sections.
i.e. if the gradient direction falls in between the angle -22.5 and 22.5, then we use the pixels that fall between this angle (r and q) as the value to compare with pixel p, see image below.



Non maximum suppression can be achieved by interpolating the pixels for greater accuracy:
$r=\alpha b+(1-\alpha)a$

Canny Edge Detection Example

Gradient Magnitude

$$|G| = \sqrt{G_x^2 + G_y^2}$$

Gradient Direction

$$\angle G = arctan(G_y / G_x)$$

Input Image: Lena

Grey Scale

Gaussian Blur

G_x

G_y

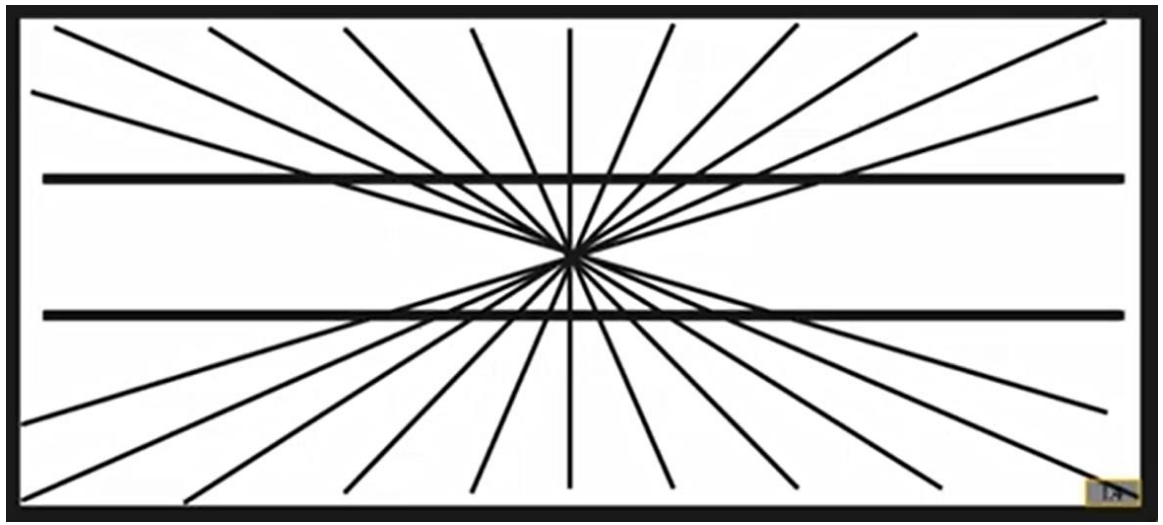
Gradient Magnitude

Non Maximum Suppression with Interpolation

Edge Thresholding + Tracking

Output Image

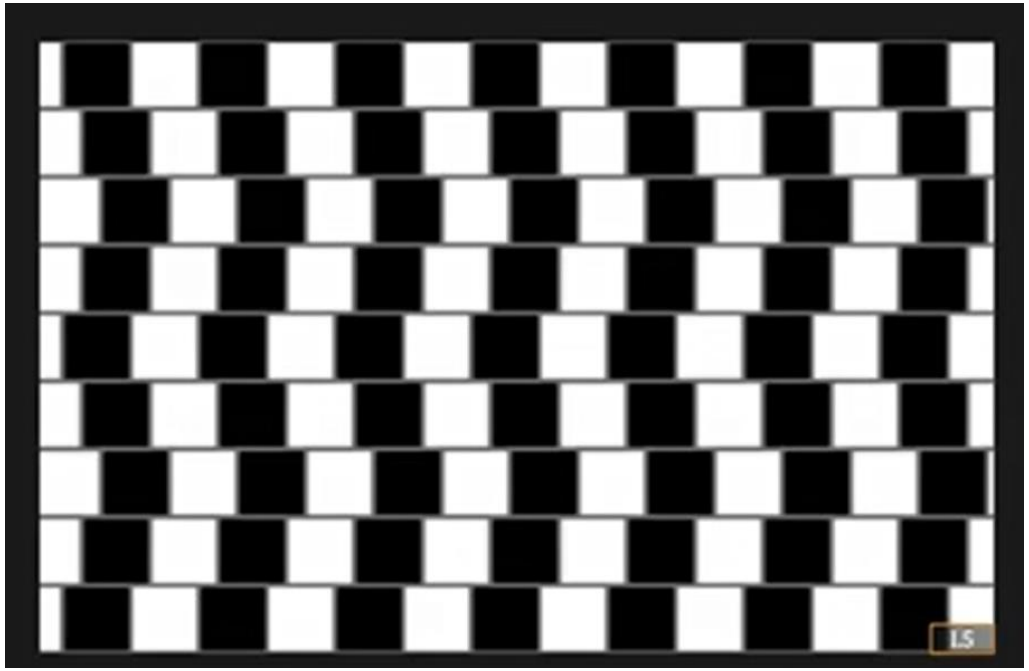
Edge Illusions: Hering Illusion

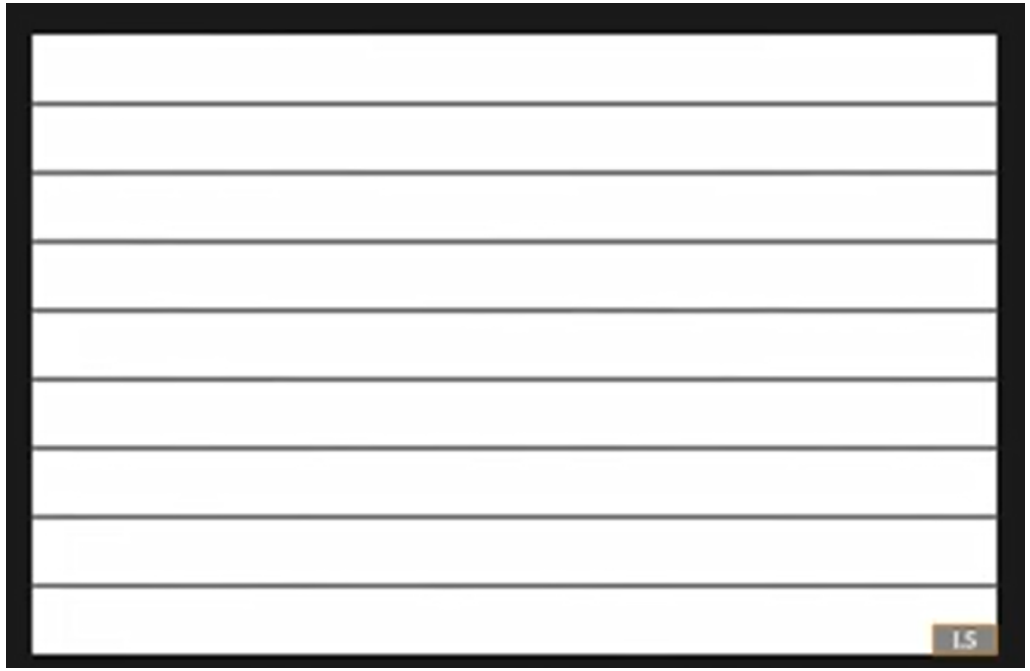


Edward Hering, 1861

Note: Human Visual systems tends to see acute angles less Acute

Edge illusions: Café Wall Illusion





Gregory and Heard, 1979

Note: Human Visual systems tends to see black patches as smaller than white patches

EYE and BRAIN

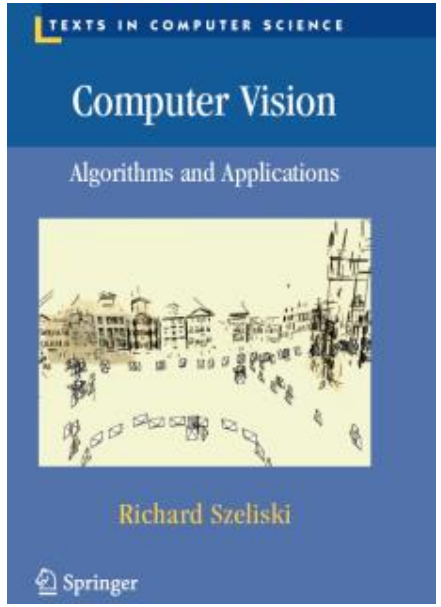
Text Books and References

❖Text Books

- Richard Szeliski. ”Computer vision: Algorithms and Applications. Springer Nature, Second Edition”, 2022
- E. R. Davies, Computer Vision Principles, Algorithms, Applications, Learning, Elsevier,5th Edition, 2017

❖ References

- Rafael C. Gonzales, Richard E. Woods, “Digital Image Processing”, Fourth Edition, Pearson Education, 2018.
- Richard Szeliski ,”Computer Vision: Algorithms and Applications” , Springer 2015
- Intro to Digital Image Processing
- Digital Image Processing 2nd edition



PDF online

❖ Credits: Slides and Video content borrowed from:

- First Principles of Computer Vision
- https://robotacademy.net.au/
- **MIT 6.S094: Computer Vision**
- **MIT Introduction to Deep Learning | 6.S191**
- Digital Image Processing 2nd edition
- Penn State University CSE/EE486 Computer Vision I
- https://justin-liang.com/tutorials/canny/#suppression (Canny Edge Detection example)

Additional

Example

original image

Demo: http://bigwww.epfl.ch/demo/ip/demos/edgeDetector/

Image credit: Joseph Redmon

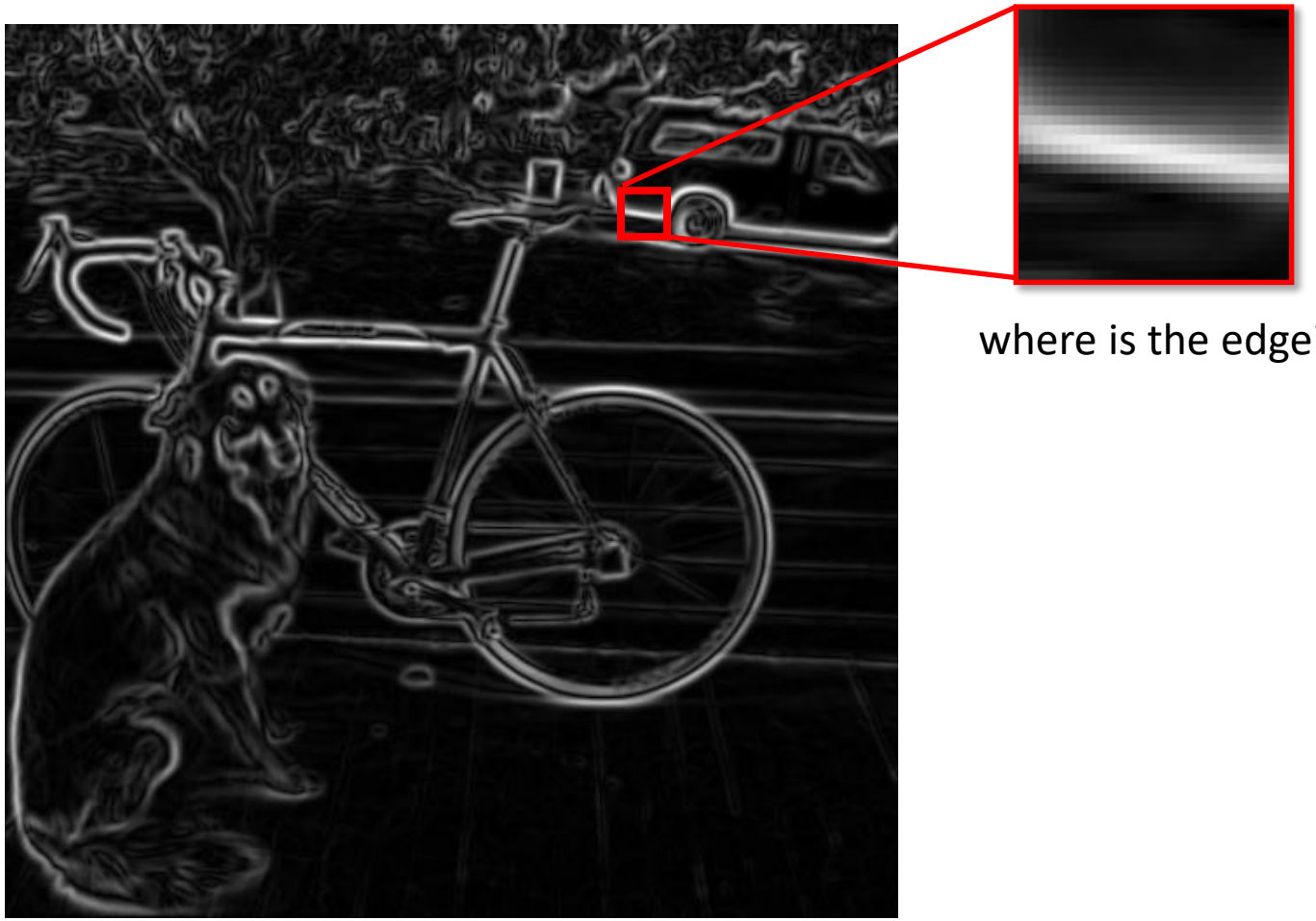
Finding edges

smoothed gradient magnitude

Finding edges

smoothed gradient magnitude

Finding edges

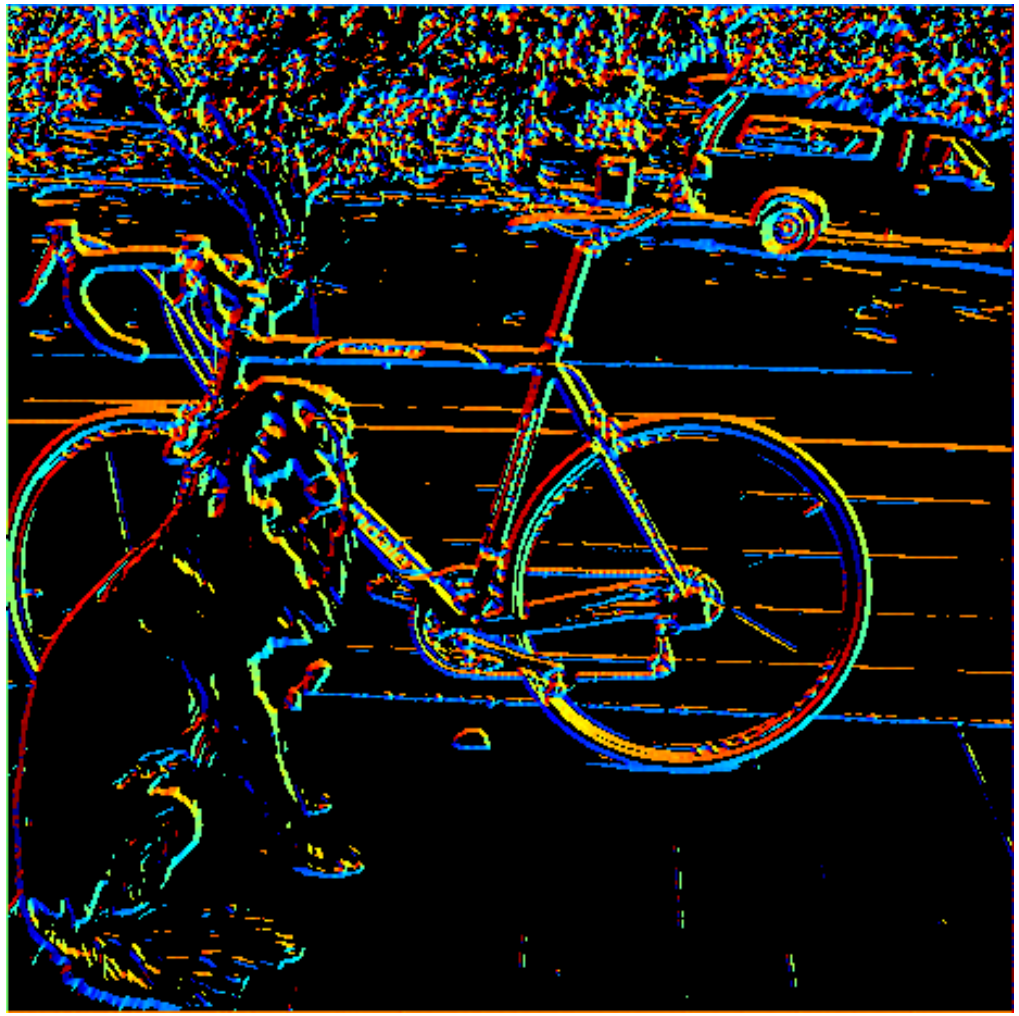


where is the edge?

thresholding

Get Orientation at Each Pixel

❖Get orientation (below, threshold at minimum gradient magnitude)



theta = atan2(gy, gx)

Gradient orientation angle

Before Non-max Suppression

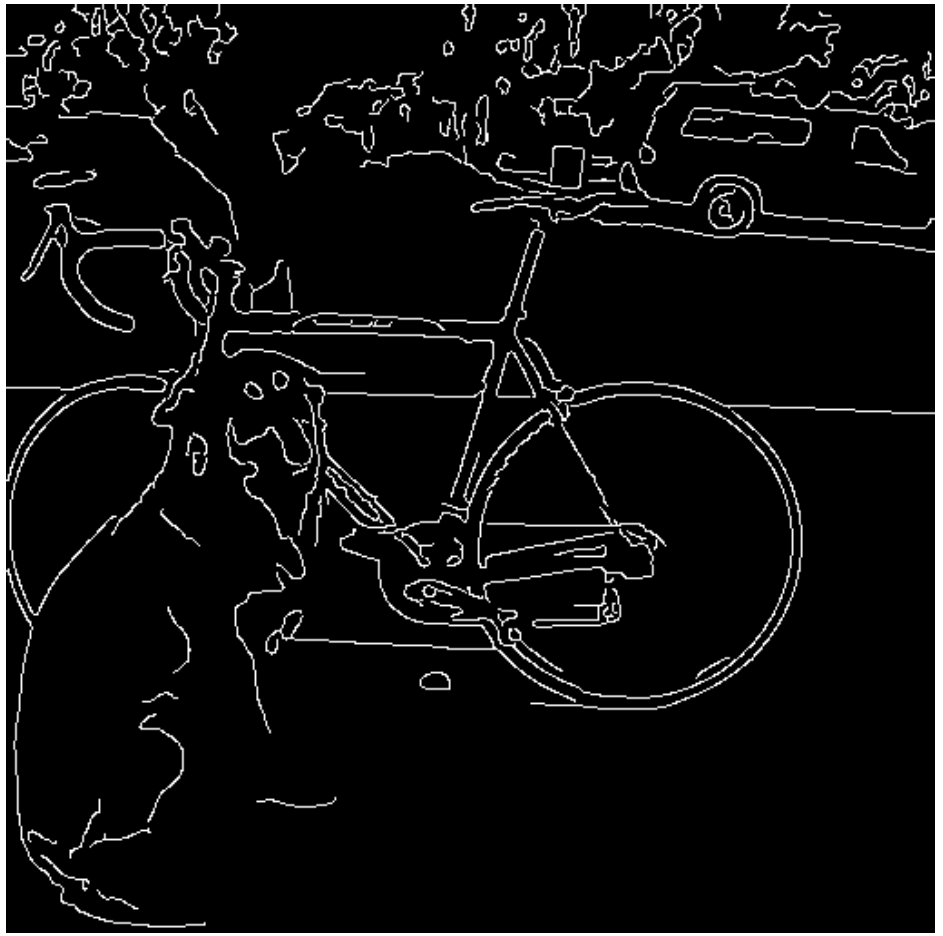
After Non-max Suppression

Thresholding edges

- Still some noise
- Only want strong edges
- 2 thresholds, 3 cases
  - R > T: strong edge
  - R < T but R > t: weak edge
  - R < t: no edge
- Why two thresholds?

Connecting edges

- Strong edges are edges!
- Weak edges are edges
  iff they connect to strong
- Look in some neighborhood
  (usually 8 closest)



Canny edge detector

MATLAB: `edge(image,'canny')`

1. Filter image with derivative of Gaussian
2. Find magnitude and orientation of gradient
3. Non-maximum suppression
4. Linking and thresholding (hysteresis):
   - Define two thresholds: low and high
   - Use the high threshold to start edge curves and the low threshold to continue them

Source: D. Lowe, L. Fei-Fei, J. Redmon

Canny edge detector

- Our first computer vision pipeline!
- Still a widely used edge detector in computer vision

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

- Depends on several parameters:

high threshold
low threshold

σ : width of the Gaussian blur

Canny edge detector

original

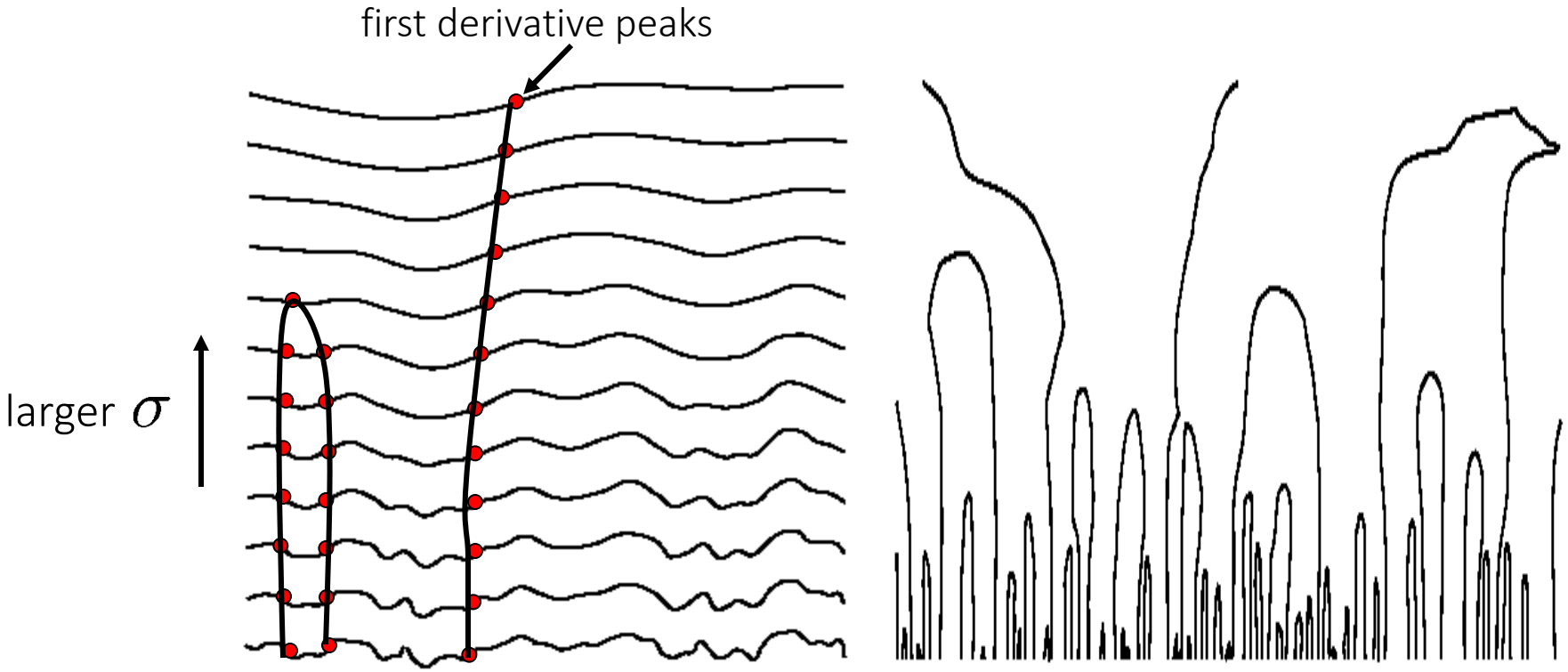
Canny with $\sigma = 1$

Canny with $\sigma = 2$

- The choice of σ depends on desired behavior
  - large σ detects "large-scale" edges
  - small σ detects fine edges

Source: S. Seitz

Scale space [Witkin 83]



❖ Properties of scale space (w/ Gaussian smoothing)

- edge position may shift with increasing scale (σ)
- two edges may merge with increasing scale
- an edge may ***not*** split into two with increasing scale

Reference: http://www.sci.utah.edu/~gerig/CS7960-S2010/materials/Witkin/Witkin-long-version-opt.pdf