Edge Detection

Computer Vision

Aziz M. Qaroush

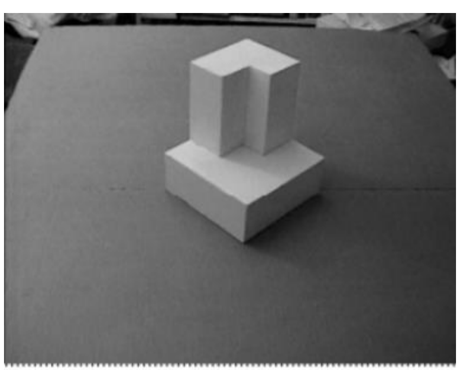
Birzeit University

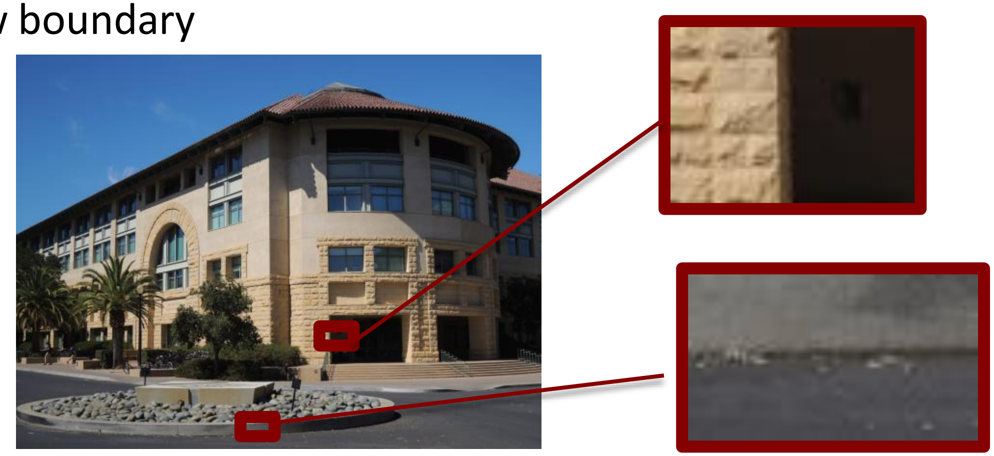
Outline

- Introduction
- Edge Detection Using Gradient
- Edge Detection Using Laplacian
- Effect of Noise on Edge Detection
- Canny Algorithm
- Other Approaches

Edges

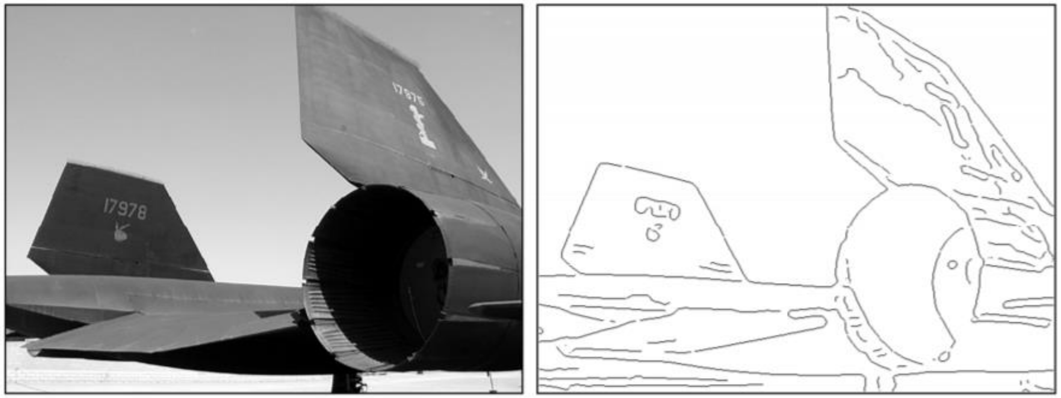
- Edges are sharp change in brightness (discontinuities).
- Edges are significant local changes of intensity in an image.
- Where do edges occur?
  - **Actual edges**: Boundaries between objects
  - Sharp change in brightness can also occur within object
    - Reflectance changes
    - Change in surface orientation
    - Illumination changes. E.g. Cast shadow boundary





Edge detection

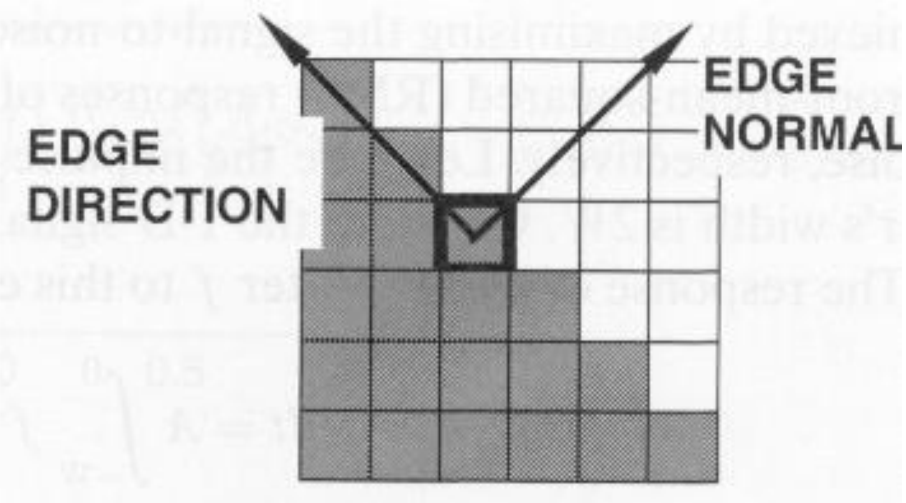
- Edge detection is the process of finding edges and contours in images
- Why do we care about edges?
  - Important features can be extracted from the edges
  - Further processing of edges into lines, curves and circular arcs result in useful features for matching and recognition.
  - Recover geometry and viewpoint



(a) (b)

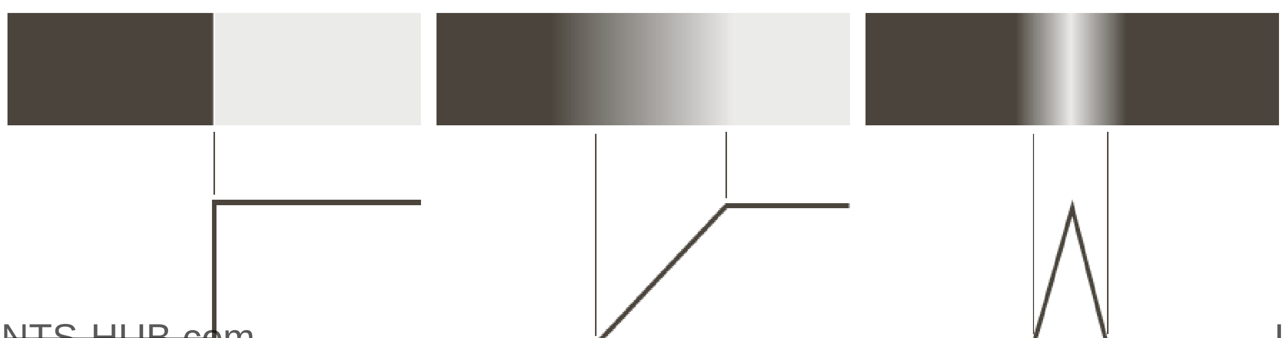
Edge Descriptors

- **Edge direction:** perpendicular to the direction of maximum intensity change (i.e., edge normal)
- **Edge strength:** related to the local image contrast along the normal.
- **Edge position:** the image position at which the edge is located.



Types of Edge

- **Edges can be classified according to intensity profiles into**
  - Step Edge (ideal edge)
    - *involves transition between two intensity levels over a distance of one pixel*
    - *Occur mostly in computer generated images*
  - *Ramp Edge*
    - *The transition between two intensity levels occur over a distance that is greater than one pixel*
    - *Appear in real images as a result of noise and focusing limitations of imaging devices*
  - *Roof Edge*
    - *Essentially, they represent blurred lines that pass through a region*

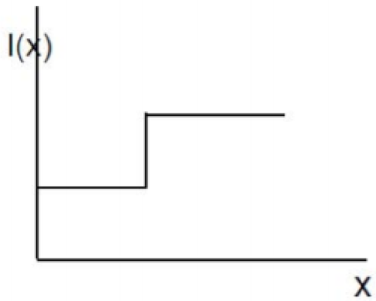


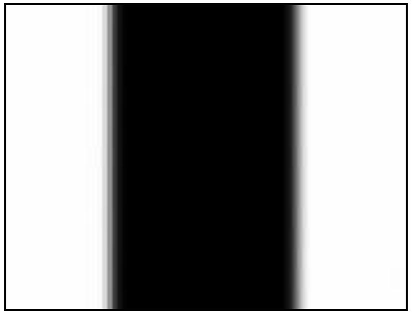
a b c

FIGURE 10.8 From left to right, models (ideal representations) of a step, a ramp, and a roof edge, and their corresponding intensity profiles.

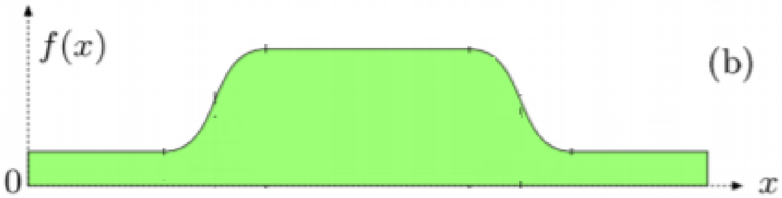
Characteristics of an Edge

- Ideal edge is a step function in a certain direction



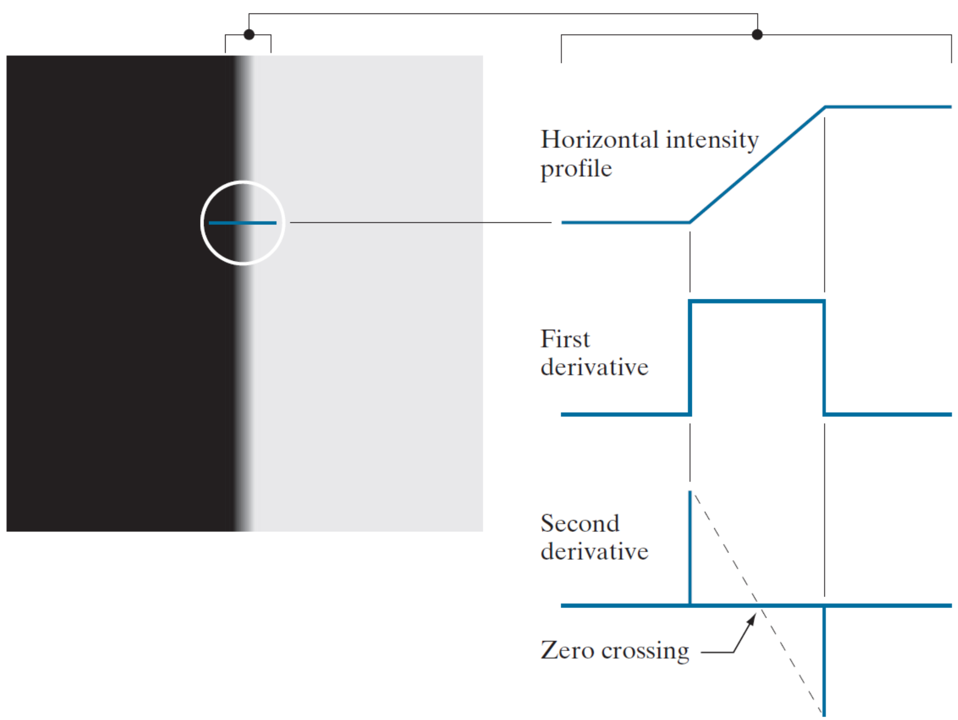


- Real (non-ideal) edge is a slightly blurred step function

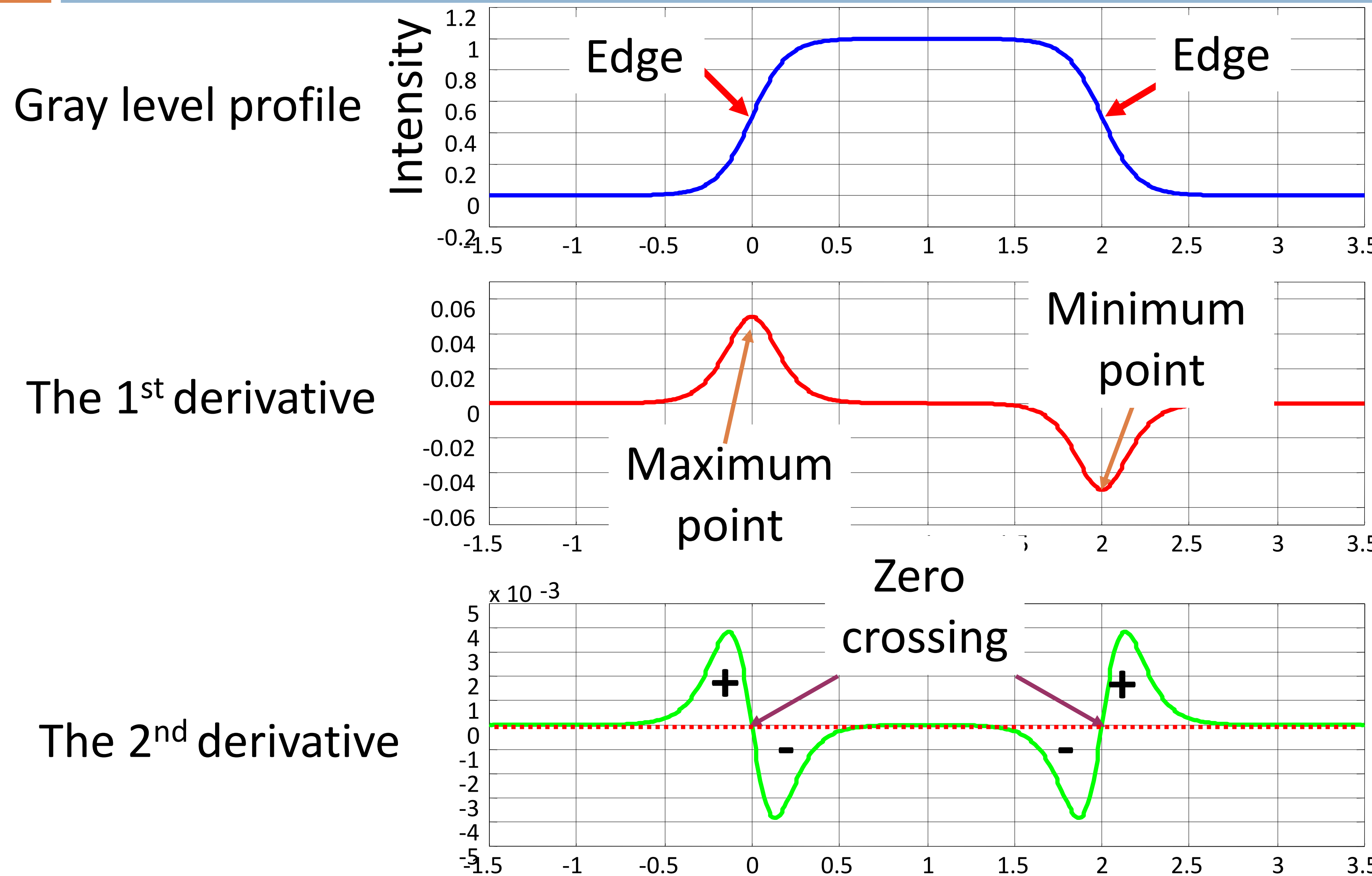


Basic Edge Detection Methods

- Detection of edges can be through the use of first-order or second-order derivatives
  - *For the first derivative, the magnitude can be used to detect the presence of an edge*
  - *For the second derivative, the sign of the second derivative is used to detect the presence of the edge*



Basic Edge Detection Methods

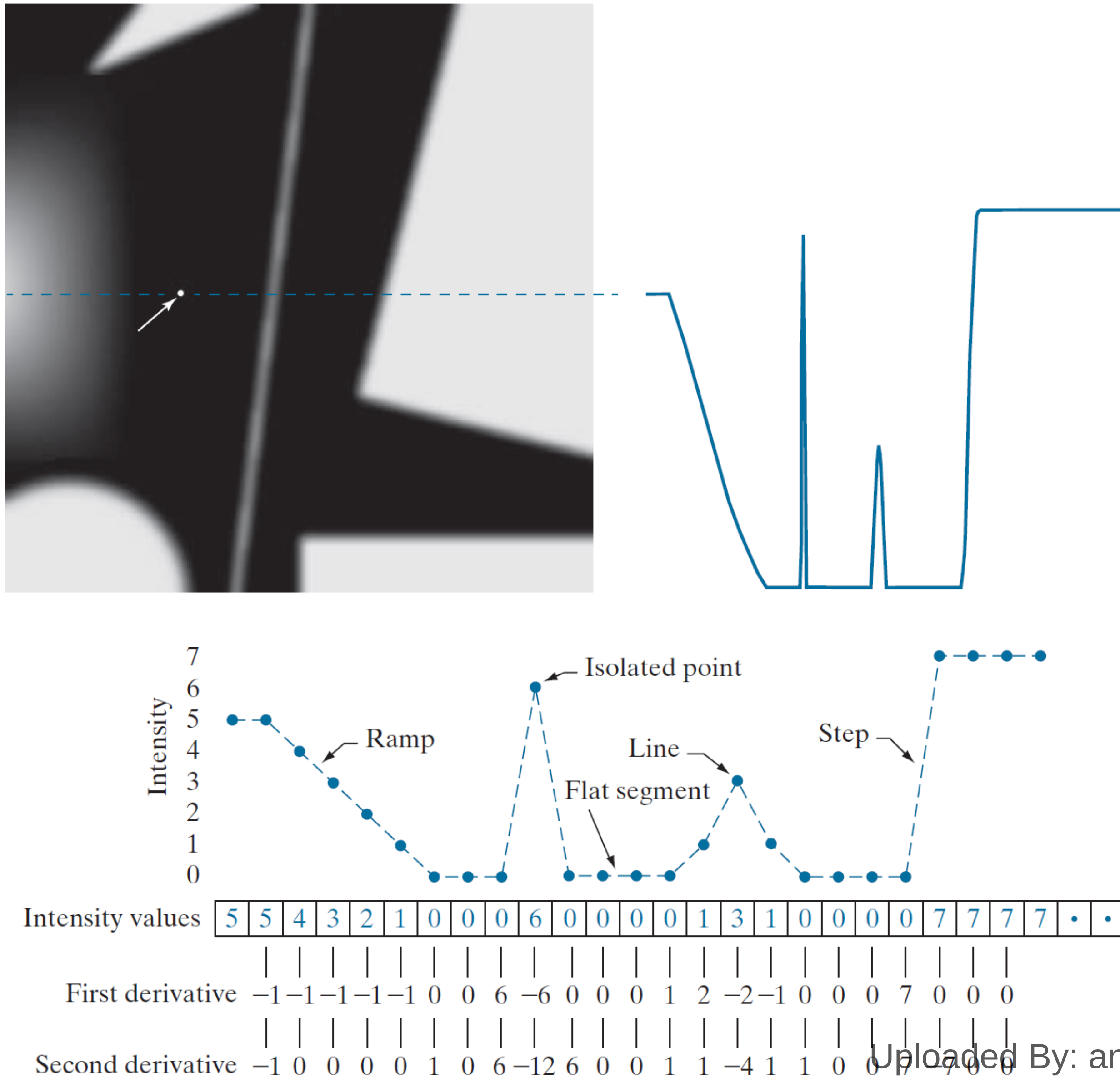


Therefore, for detecting edges, we can apply zero crossing detection to the 2nd derivative image or thresholding the absolute of the 1st derivative image

Edge Detection: derivatives behavior

a b
c

FIGURE 10.2
(a) Image.
(b) Horizontal intensity profile that includes the isolated point indicated by the arrow.
(c) Subsampled profile; the dashes were added for clarity. The numbers in the boxes are the intensity values of the dots shown in the profile. The derivatives were obtained using Eqs. (10-4) for the first derivative and Eq. (10-7) for the second.



Edge Detection: derivatives behavior

- We are concerned about the behavior of 1^{st} and 2^{nd} derivatives in the following areas
  - Constant intensity
  - Onset and end of discontinuities (ramps and steps)
  - Intensity ramps
- Properties of 1^{st} derivative
  - Zero in areas of constant intensity
  - Nonzero at the onset of a step and intensity ramp
  - Nonzero along intensity ramp
- Properties 2^{nd} derivative
  - Zero in areas of constant intensity
  - Nonzero at the onset and end of a step and intensity ramp
  - Zero along intensity ramp

Edge Detection: derivatives behavior

- **Notes**
  - Examining the 1st and 2nd derivatives plots shows that all of their properties are satisfied
  - 1st derivative produce thicker edges than 2nd derivatives
  - 2nd derivatives have a stronger response to fine detail, such as thin lines, isolated points, and noise.
  - 2nd derivative produce double edge separated by a zero crossing
  - The sign of the 2nd derivative can be used to determine whether a transition into an edge is from light to dark or dark to light.
  - 1st derivative is commonly used in edge detection since:
    - Less sensitive to fine details.
    - The first derivative is often less sensitive to noise compared to the second derivative.
    - Provides information about both the magnitude and direction of intensity changes.
    - less computationally expensive to calculate.

Edge detection methods

- First order derivative
  - Roberts operator
  - Prewitt operator
  - Sobel operator
  - Cany edge detector (optimal algorithm)
- Second order derivative
  - Laplacian of Gaussian
  - Difference of Gaussian
  - Marr-Hildreth (LoG-based) Edge Detector

Edge Detection Using Gradient (First Derivative)

- The gradient is a powerful tool in finding the strength and direction of edges. The gradient at pixel (x,y) is defined as

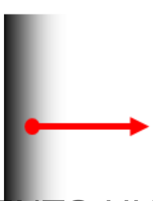
$$\nabla \mathrm{f} = \begin{bmatrix} G_x \\ G_y \end{bmatrix} = \begin{bmatrix} \dfrac{\partial f}{\partial x} \\ \dfrac{\partial f}{\partial y} \end{bmatrix}$$

- The magnitude of the gradient measures the strength of the edge (maximum rate of change)

$$M(x,y) = \left[\left(\frac{\partial f}{\partial x} \right)^2 + \left(\frac{\partial f}{\partial y} \right)^2 \right]^{1/2} \qquad |\nabla f| \approx |G_x| + |G_y|$$

- The direction of the gradient is perpendicular to the edge direction

$$\alpha = \tan^{-1}\left(\frac{G_y}{G_x}\right)$$

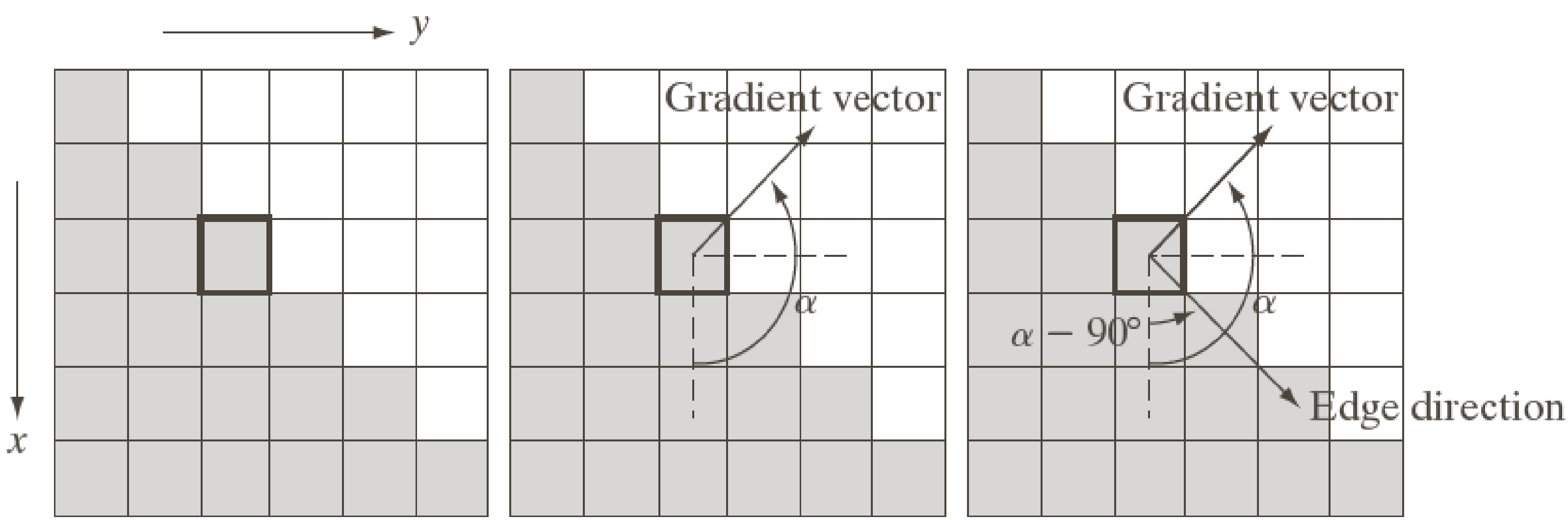


$$\nabla f = \left[\frac{\partial f}{\partial x}, 0\right]$$

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

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

Edge Detection Using Gradient



$$\nabla \mathrm{f} = \begin{bmatrix} G_x \\ G_y \end{bmatrix} = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix} = \begin{bmatrix} -2 \\ 2 \end{bmatrix}$$

$$\alpha = \tan^{-1}\left(\frac{G_y}{G_x}\right) = 135^o$$

- All edge pixels have the same gradient magnitude and direction

Edge Detection Using Gradient

- **Gradient Masks to compute gradient at Z_5**
  - **Discrete 1st Derivative**

$$G_x = \frac{\partial f}{\partial x} = f(x+1,y) - f(x,y)$$
$$= Z_8 - Z_5$$

-1
1

$$G_y = \frac{\partial f}{\partial y} = f(x,y+1) - f(x,y)$$
$$= Z_6 - Z_5$$

-1	1

- Not efficient in detecting diagonal edges

  - **Roberts Cross-gradient Operator**

-1	0
0	1

Horizontal Operator

0	-1
1	0

Vertical Operator

$$G_x(x,y) = z_9 - z_5$$
$$G_y(x,y) = z_8 - z_6$$

Z1	Z2	Z3
Z4	Z5	Z6
Z7	Z8	Z9

Pixel z5 and its neighbours

  - 2x2 masks are not as good as symmetric masks which capture information from opposite sides around the center point

Edge Detection Using Gradient

- **Gradient Masks to compute gradient at Z_5**

Z1	Z2	Z3
Z4	Z5	Z6
Z7	Z8	Z9

Pixel z5 and its neighbours

  - **Prewitt Operators**

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

Mask to Compute Gx

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

Mask to Compute Gy

$$G_x = \frac{\partial f}{\partial x} = (Z_7 + Z_8 + Z_9) - (Z_1 + Z_2 + Z_3)$$

$$G_y = \frac{\partial f}{\partial y} = (Z_3 + Z_6 + Z_9) - (Z_1 + Z_4 + Z_7)$$

  - **Sobel Operators**

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

Mask to Compute Gx

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

Mask to Compute Gy

$$G_x = \frac{\partial f}{\partial x} = (Z_7 + 2Z_8 + Z_9) - (Z_1 + 2Z_2 + Z_3)$$

$$G_y = \frac{\partial f}{\partial y} = (Z_3 + 2Z_6 + Z_9) - (Z_1 + 2Z_4 + Z_7)$$

  - They have better response than Prewitt masks and have better smoothing which is essential to reduce the effect of noise

Edge Detection Using Gradient

Image

Gx computed using
Sobel Operator

Gy computed using
Sobel Operator

|Gx| + |Gy|

Angle of Gradient

Edge Detection Using Gradient

- **Gradient Masks to compute gradient at Z_5**
  - Prewitt and Sobel masks shown before give the strongest response for horizental and vertical edges. We can modify these masks to obtain better response for diagonal edges **Prewitt Operators**

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

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

Prewitt Diagonal Masks

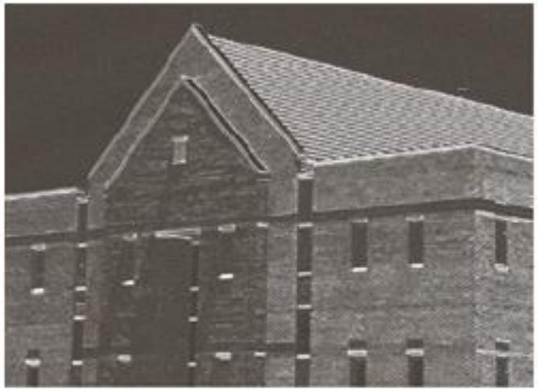
0	1	2
-1	0	1
-2	-1	0

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

Sobel Diagonal Masks

Edge Detection Using Gradient

- **Example – continued**
  - The Sobel masks used in the previous slides were those that have stronger response for vertical and horizontal directions. How about diagonal directions?



Gx computed using Sobel Operator (Horizental)

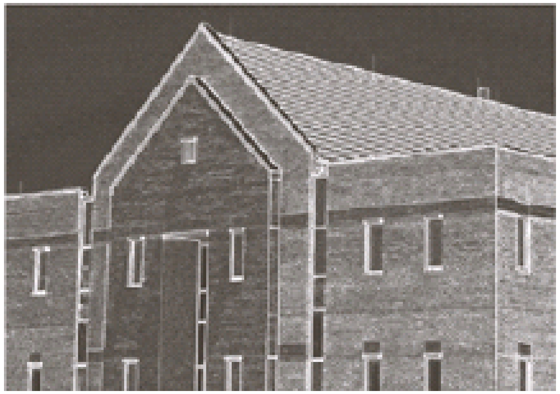
Gy computed using Sobel Operator (vertical)

Gx computed using Diagonal Sobel Operator (+45)

Gx computed using Diagonal Sobel Operator (-45)

Edge Detection Using Gradient

- **Example – continued**
  - Note that in the previous slide, the edges of the wall bricks were successfully detected. However, this might not be desirable if we are interested in the main edges
  - We can eliminate such small edges (which might considered as noise) by
    - Smoothing the image before computing the gradient
    - Thresholding the gradient image
    - Smoothing followed by thresholding



Gradient Image (|Gx| + |Gy|) without smoothing

Gradient Image (|Gx| + |Gy|) after the original image was smoothed by 5x5 mask

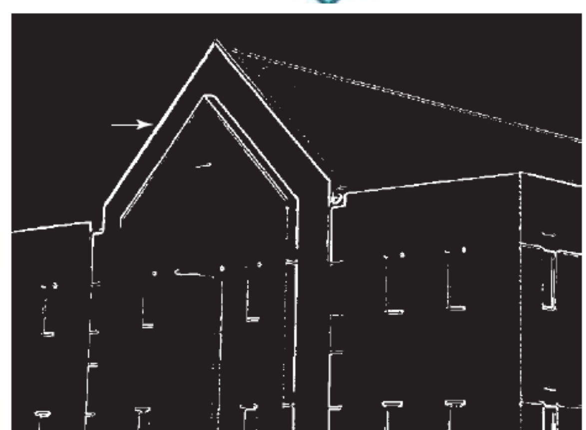
Edge Detection Using Gradient

- **Example – continued**

Gradient Image (|Gx| + |Gy|) without smoothing

Gradient Image (|Gx| + |Gy|) after the original image was smoothed by 5x5 mask

Thresholded Gradient Image



Thresholded Gradient Image (better connectivity for edges)

Edge Detection Example

Original (a)

Sobel (b)

Prewitt (c)

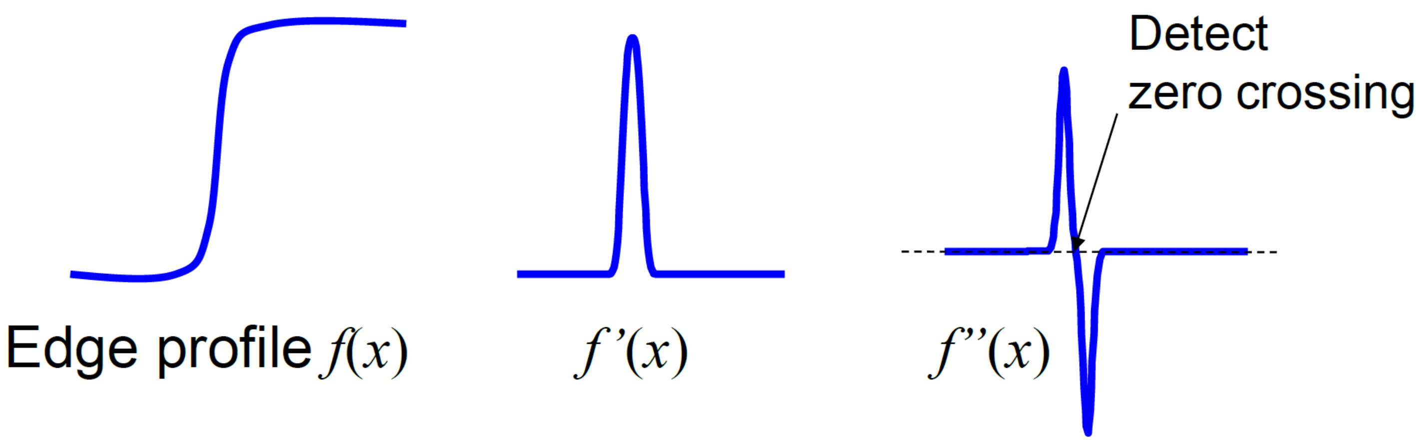
Roberts (d)

Edge detection using Laplacian operator

- Detect edges by considering second derivative

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

- Isotropic (rotationally invariant) operator
- Zero-crossings mark edge location



Edge detection using Laplacian operator

- When we consider an image function of two variables, f(x,y), at which time we will dealing with partial derivatives along the two spatial axes.
- The second derivative (Laplacian) in 2-D is defined as

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

- If we define

$$\frac{\partial^2 f}{\partial x^2} = f(x+1,y) + f(x-1,y) - 2f(x,y)$$

$$\frac{\partial^2 f}{\partial y^2} = f(x,y+1) + f(x,y-1) - 2f(x,y)$$

- Then

$$\nabla^2 f = [f(x+1,y) + f(x-1,y) + f(x,y+1) + f(x,y-1) - 4f(x,y)]$$

Edge detection using Laplacian operator

- The Laplacian can be implemented as a filter mask

0	1	0
1	-4	1
0	1	0

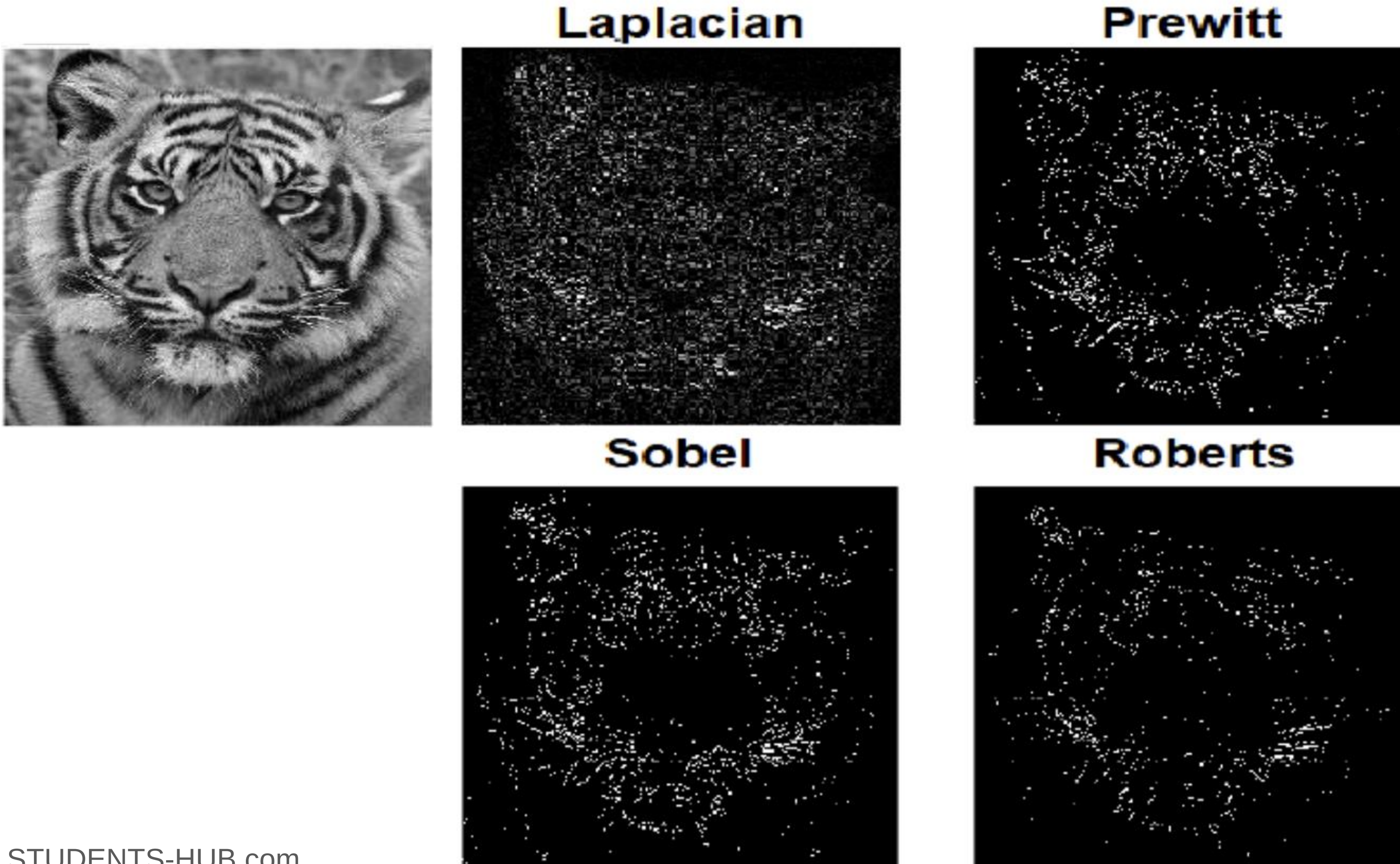
- Or

1	1	1
1	-8	1
1	1	1

Edge detection using Laplacian operator

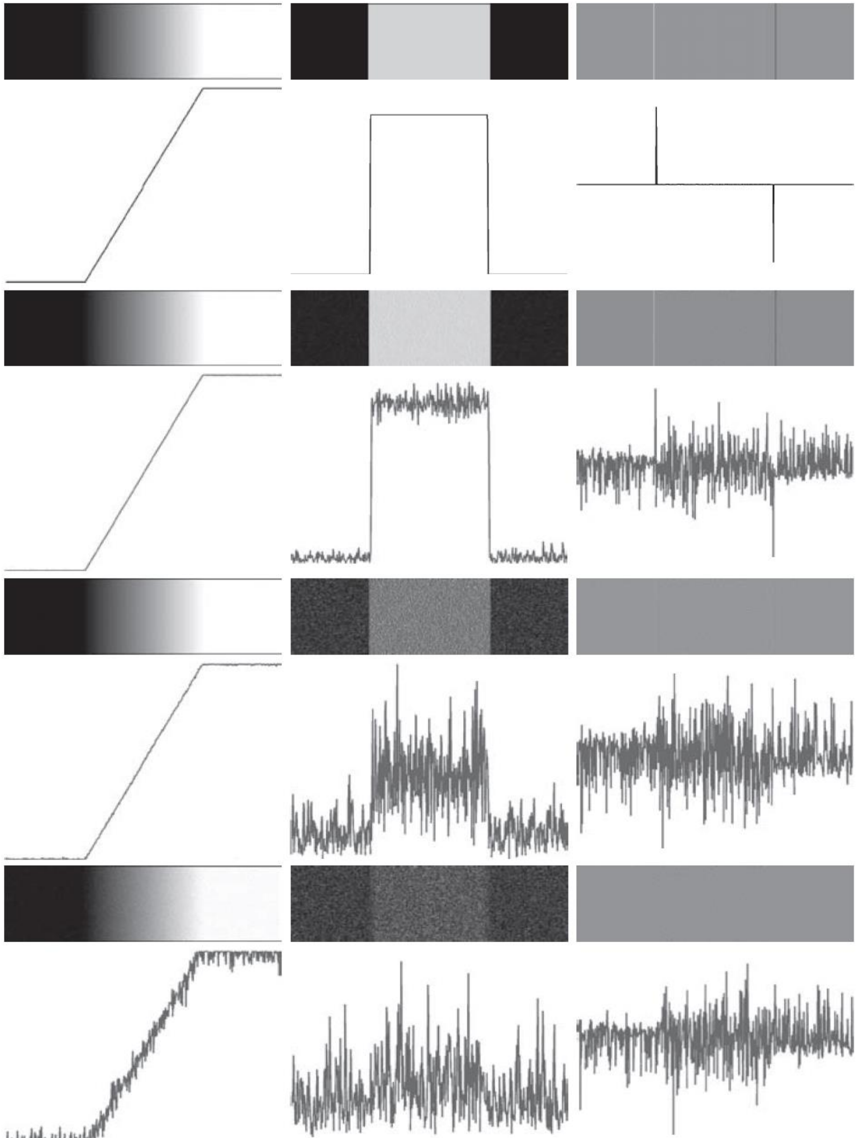
0	1	0
1	-4	1
0	1	0

Comparisons

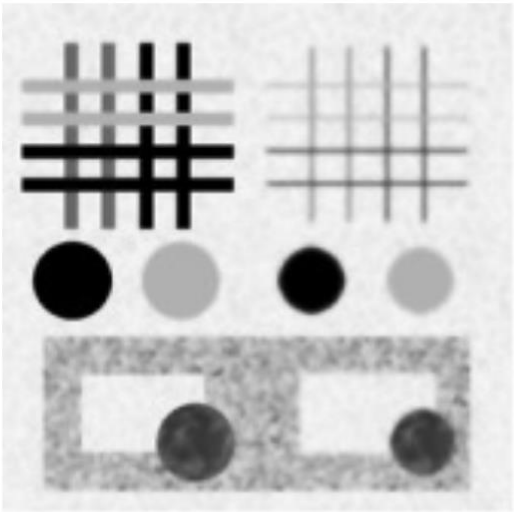


Edge Detection with the Existence of Noise

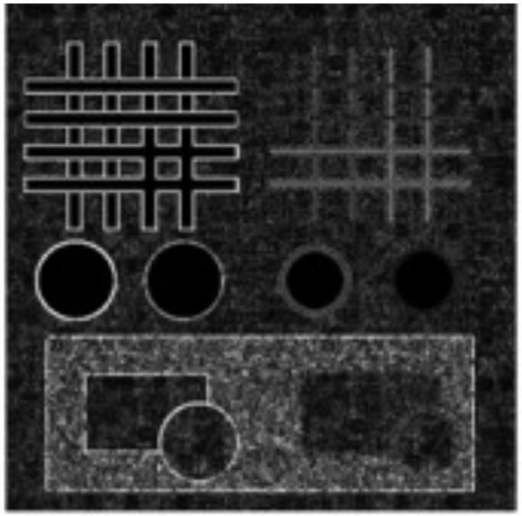
- First column: 8-bit images with values in the range [0,255], and intensity profiles of a ramp edge corrupted by Gaussian noise of zero mean and standard deviations of 0.0, 0.1, 1.0, and 10.0 intensity levels, respectively.
- Second column: First-derivative images and intensity profiles.
- Third column: Second-derivative images and intensity profiles.
- **Notes:**
  - The second derivative is more sensitive to noise.
  - It would be difficult indeed to detect edges as noise level increase.



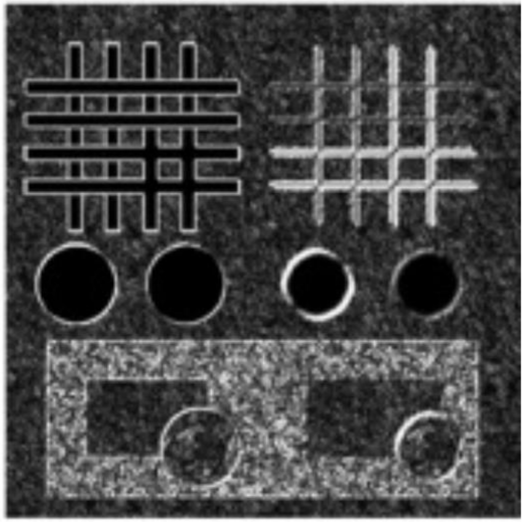
Edge Detection with Noise



Original



Laplacian



Sobel X+Y

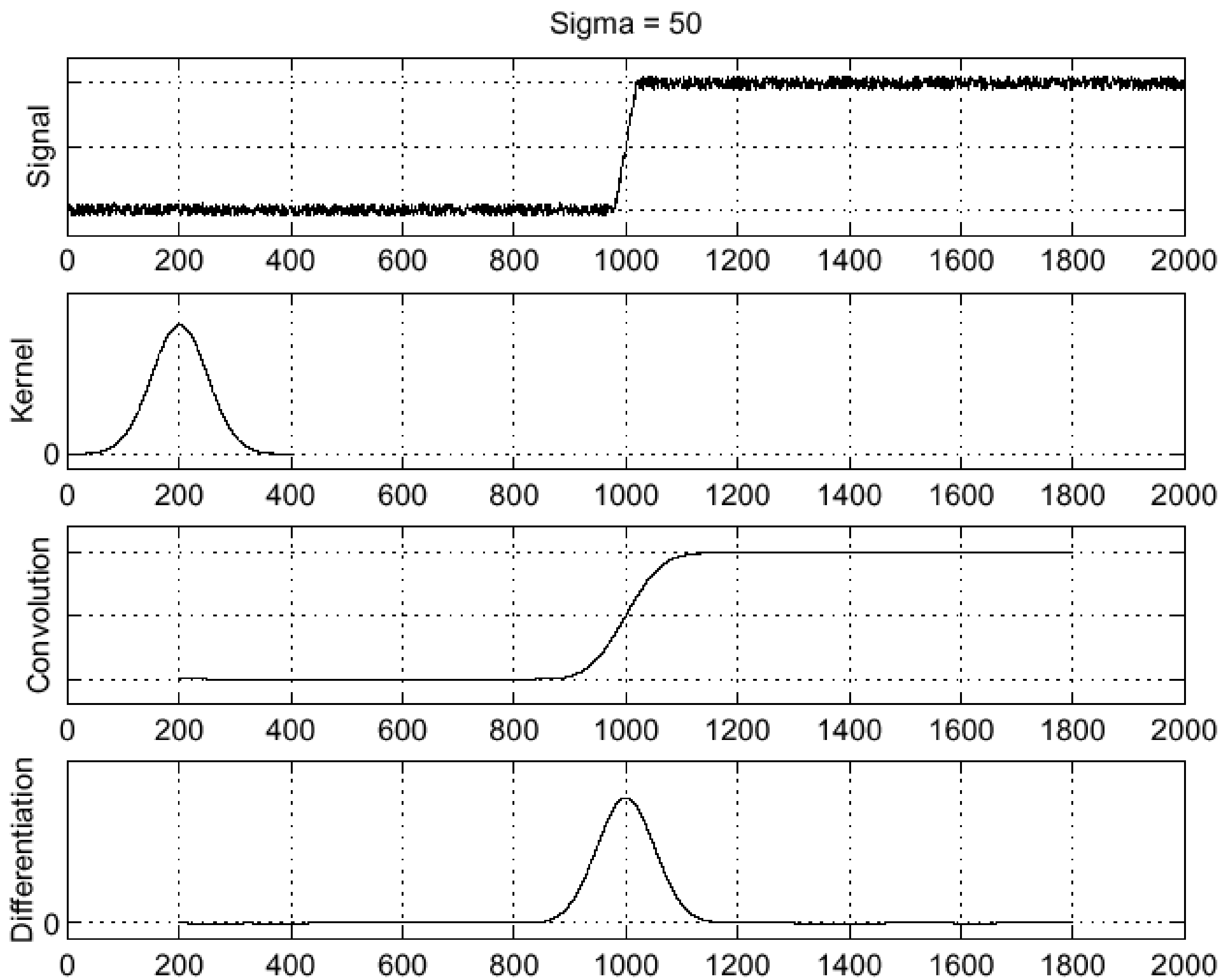
Solution: Smooth First

f

h

$h \star f$

$\frac{\partial}{\partial x}(h \star f)$



Where is the edge? Look for peaks in $\frac{\partial}{\partial x}(h \star f)$

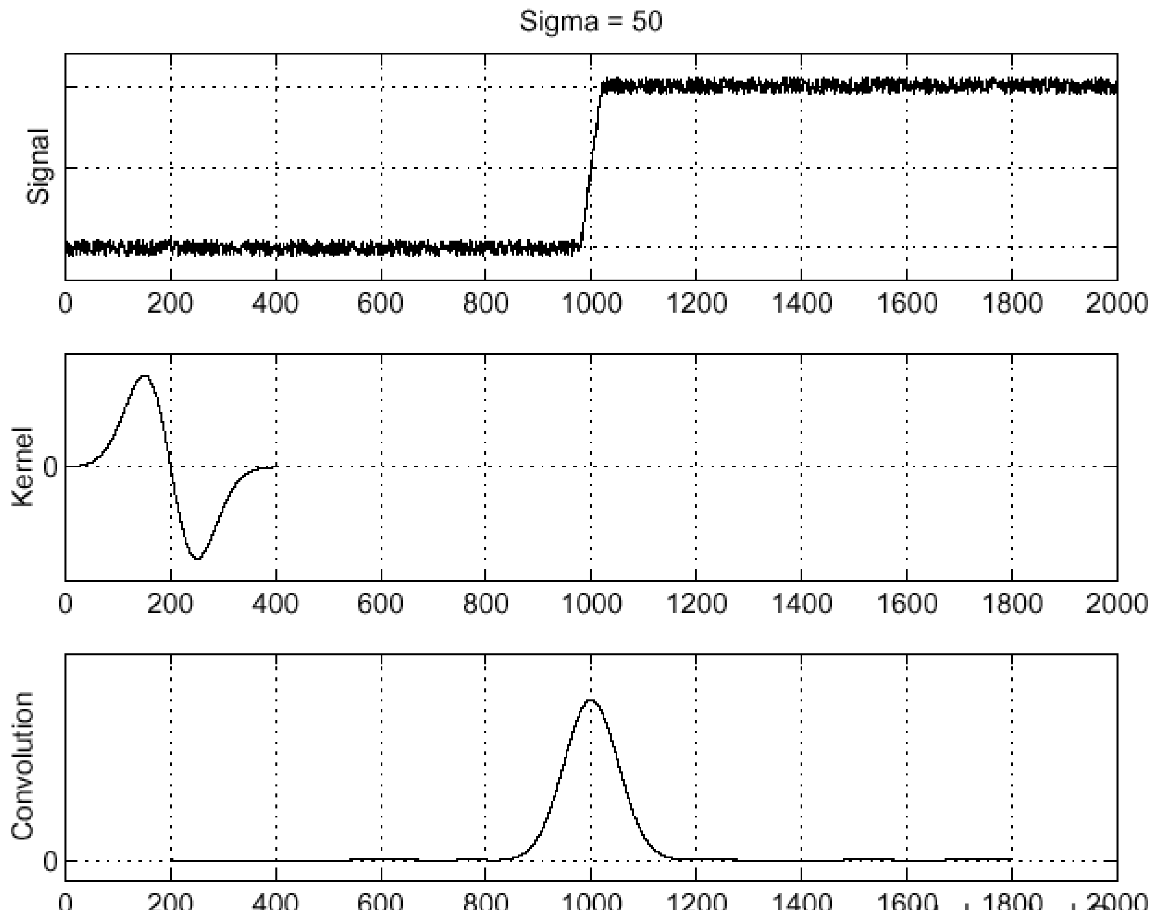
Derivative theorem of convolution

- Convolve the image with the derivative of the filter can saves us one operation: $$\frac{\partial}{\partial x}(h \star f) = (\frac{\partial}{\partial x}h) \star f$$

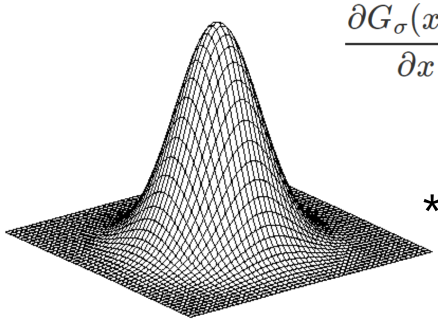
f

$\frac{\partial}{\partial x}h$

$(\frac{\partial}{\partial x}h) \star f$

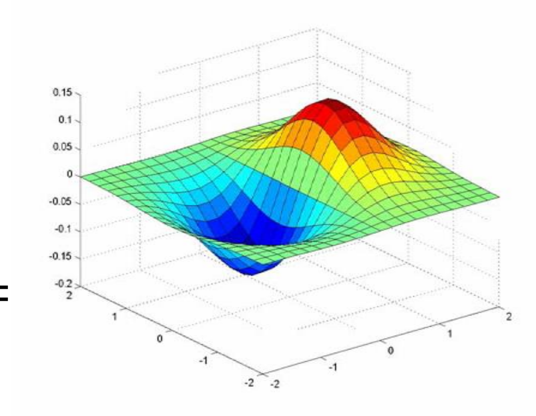


Derivative of Gaussian filter



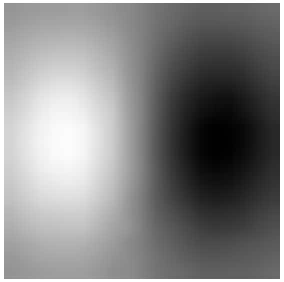
$$\frac{\partial G_\sigma(x,y)}{\partial x} = -\frac{x}{\sigma^2} G_\sigma(x,y)$$

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



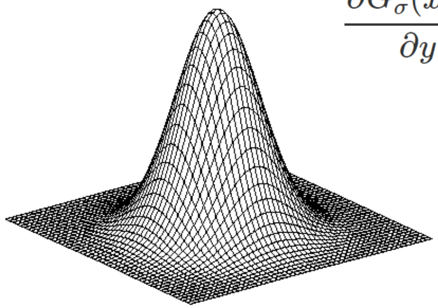
$\frac{1}{8}$

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



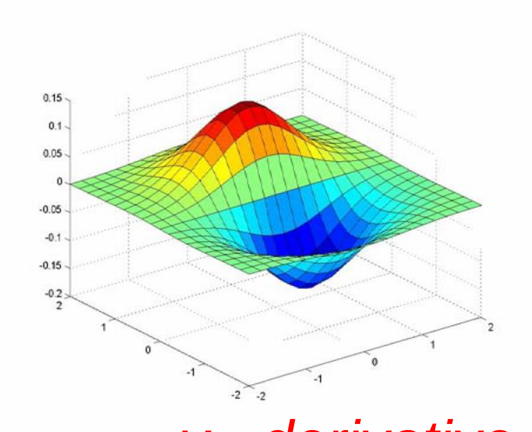
2D-gaussian

x - derivative



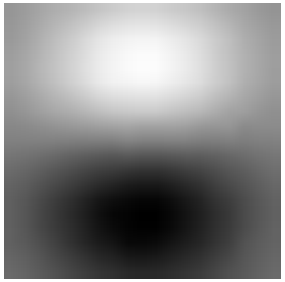
$$\frac{\partial G_\sigma(x,y)}{\partial y} = -\frac{y}{\sigma^2} G_\sigma(x,y)$$

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



$\frac{1}{8}$

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



2D-gaussian

y - derivative

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 value

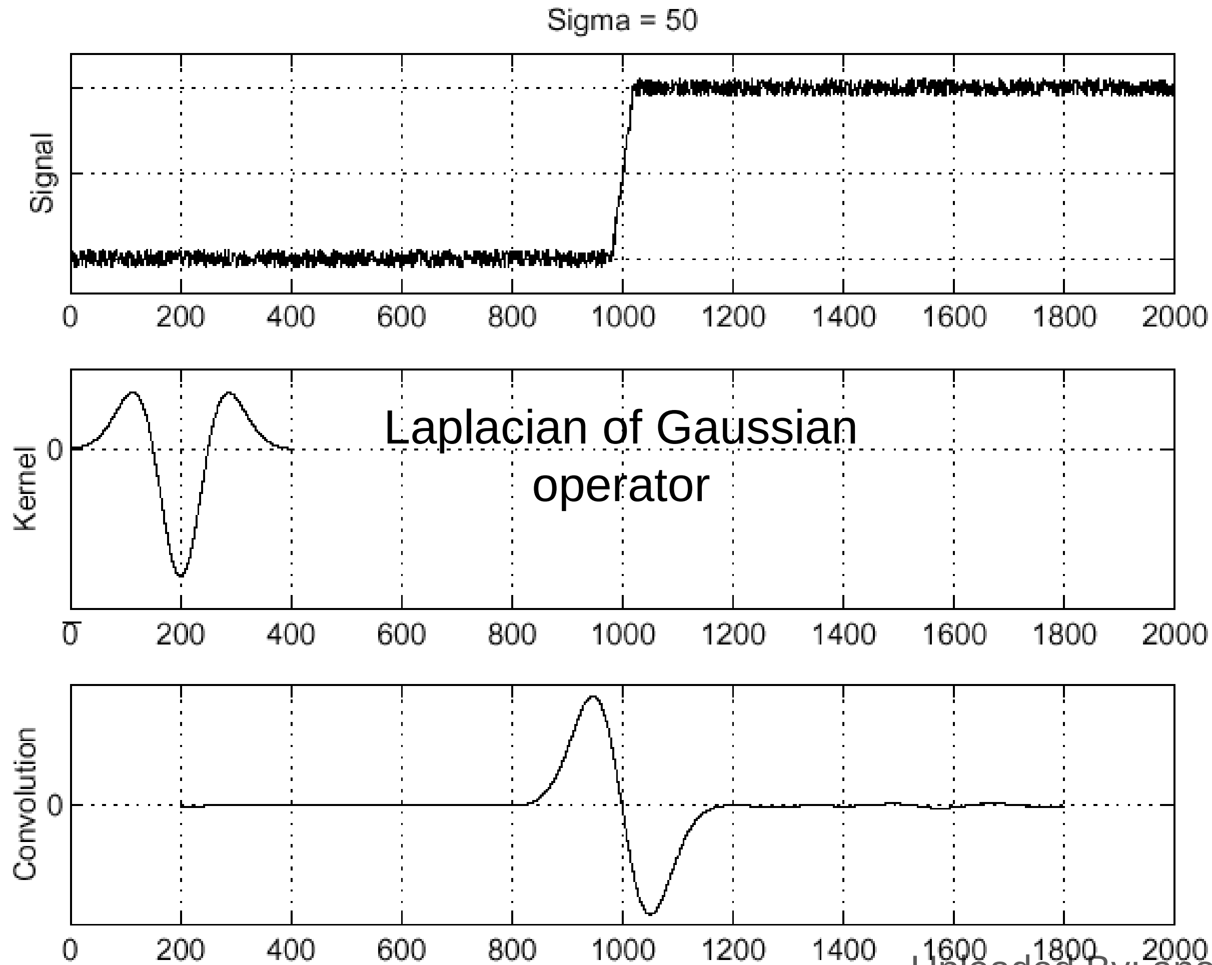
Laplacian of Gaussian

- Look for zero-crossings of $\left(\frac{\partial^2}{\partial x^2}h\right) \star f$

f

$\frac{\partial^2}{\partial x^2}h$

$\left(\frac{\partial^2}{\partial x^2}h\right) \star f$



Laplacian of Gaussian

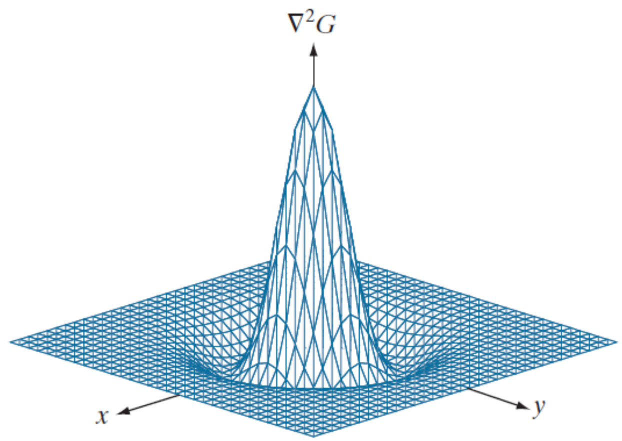
$$G(x, y; \sigma) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

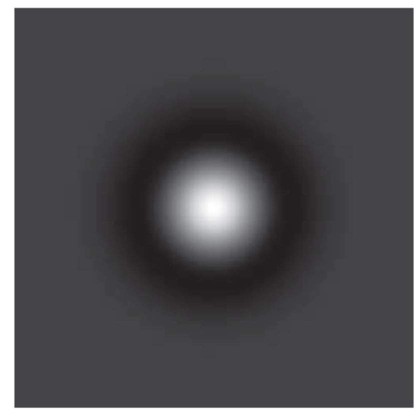
$$\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 y}{\partial y^2}$$

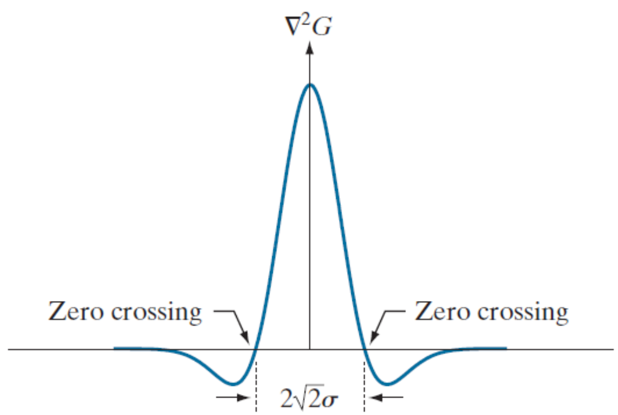
$$\nabla^2 G(x, y; \sigma) = \left(\frac{x^2 + y^2}{\sigma^4} - \frac{2}{\sigma^2} \right) G(x, y; \sigma)$$

a	b
c	d

FIGURE 10.21 (a) 3-D plot of the *negative* of the LoG. (b) Negative of the LoG displayed as an image. (c) Cross section of (a) showing zero crossings. (d) 5×5 kernel approximation to the shape in (a). The negative of this kernel would be used in practice.







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

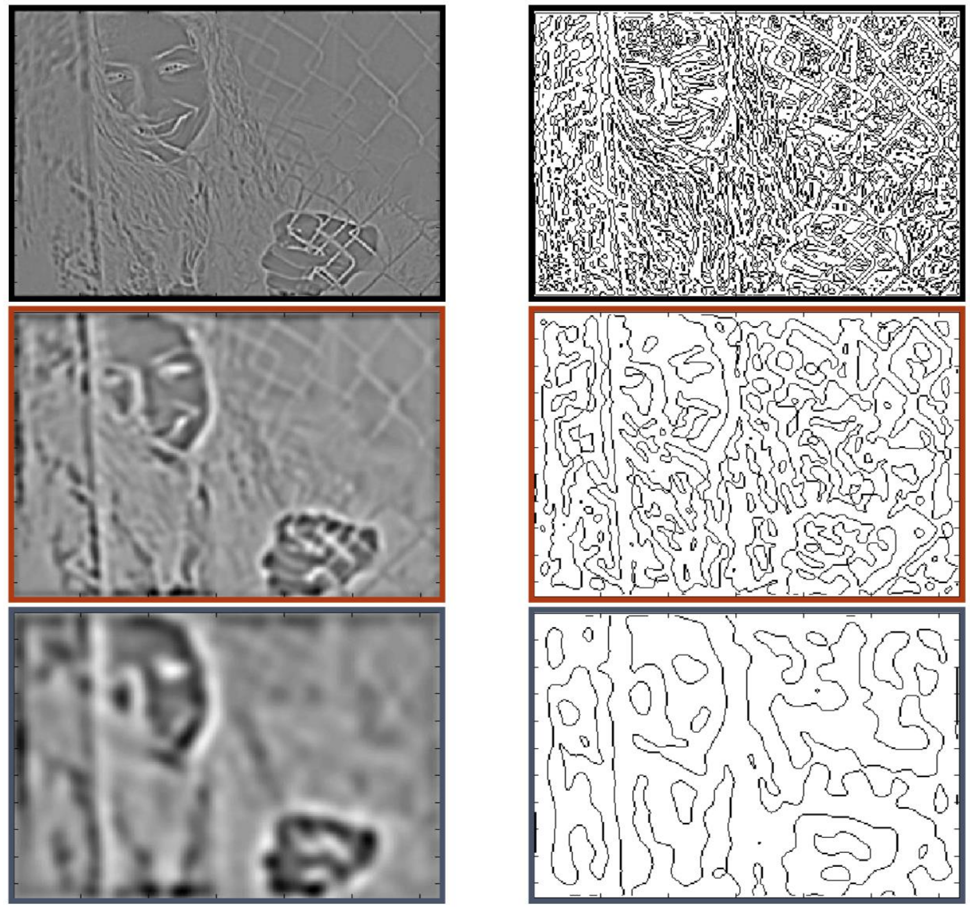
Laplacian of Gaussian

I | $I * \left(\Delta^2 g\right)$ | Zero crossings of $\Delta^2 S$

$\sigma = 1$

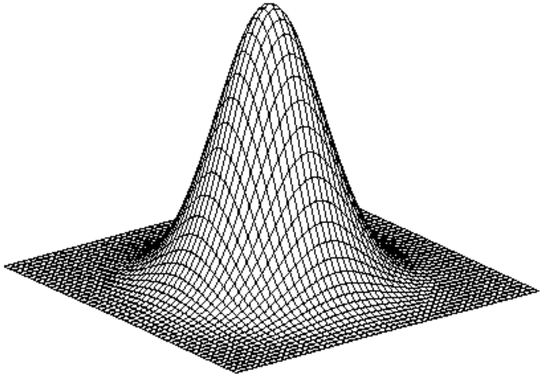
$\sigma = 3$

$\sigma = 6$

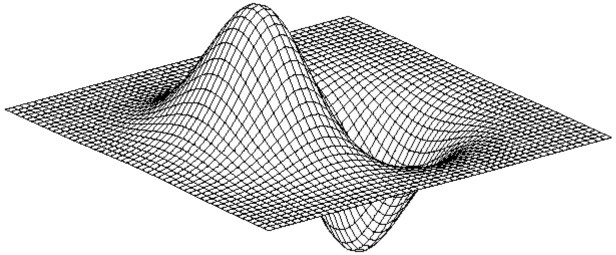


2D Edge Detection Filters

Gaussian



Derivative of Gaussian



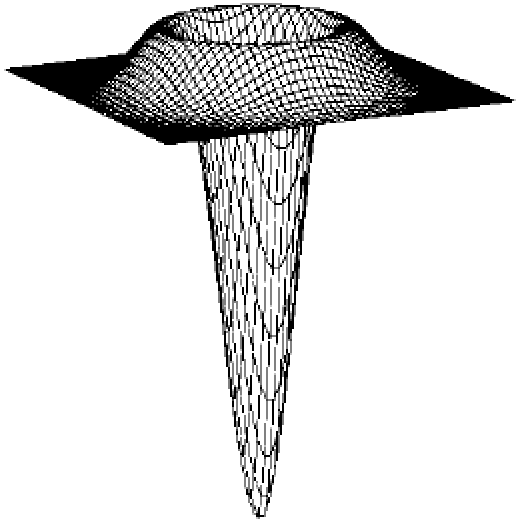
$\frac{1}{8}$

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

$\frac{1}{8}$

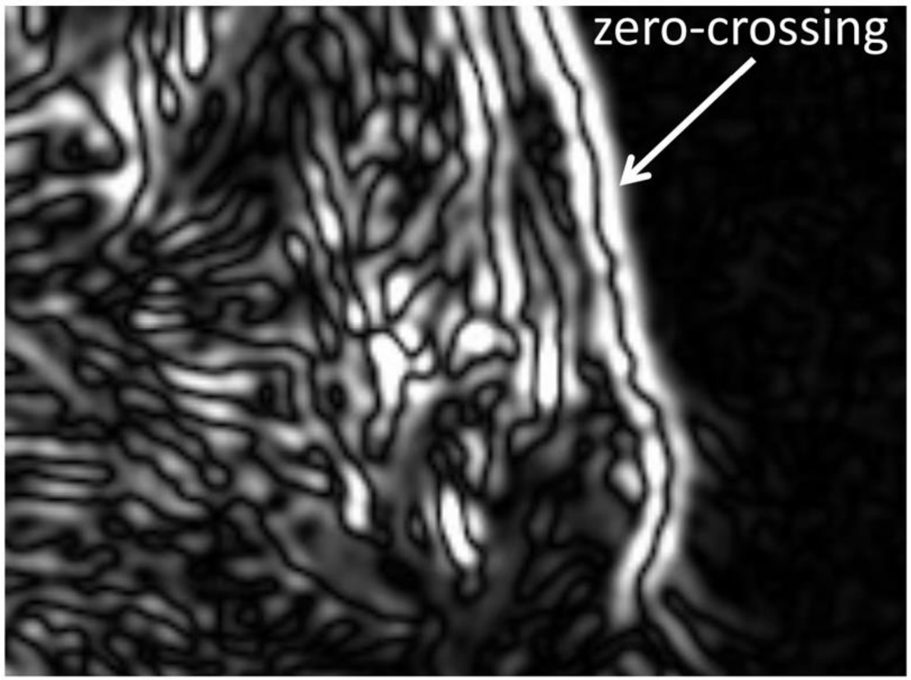
1	2	1
0	0	0
-1	-2	-1

Laplacian of Gaussian

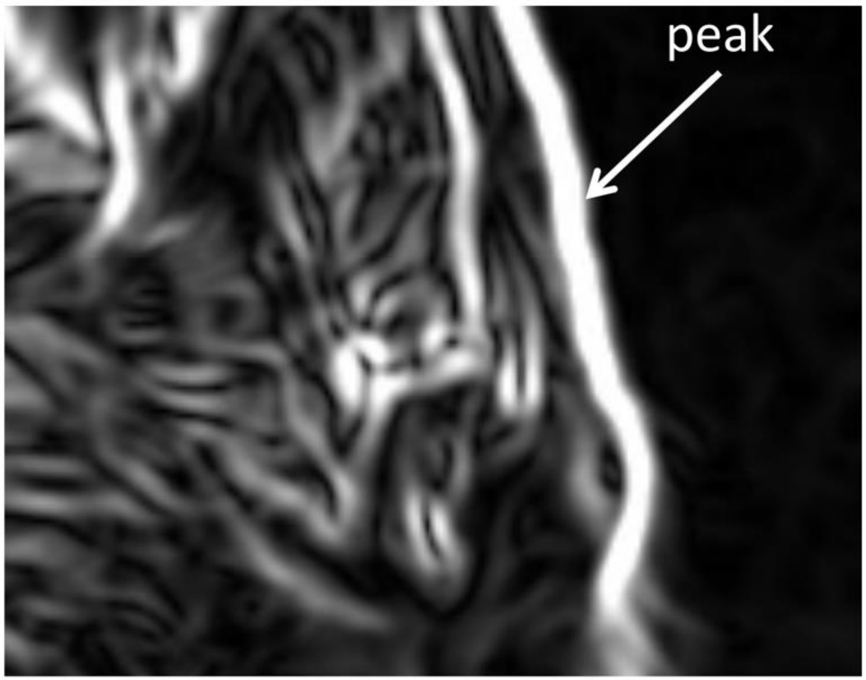


0	0	−1	0	0
0	−1	−2	−1	0
−1	−2	16	−2	−1
0	−1	−2	−1	0
0	0	−1	0	0

Laplacian of Gaussian vs Derivative of Gaussian



Laplacian of Gaussian filtering



Derivative of Gaussian filtering

Laplacian of Gaussian vs Derivative of Gaussian

Characteristic	Laplacian of Gaussian (LoG)	Derivative of Gaussian (DoG)
Type of derivative filter	Second-order derivative filter	First-order derivative filter
Convolution Kernels	Single kernel	Use separate kernels for horizontal and vertical gradients
Sensitivity to fine details	More sensitive	Less sensitive
Robustness to noise	Less robust	More robust
Type of detector	Zero-crossing detector	Peak detector
Best suited for	Edge detection in images with fine details	Edge detection in noisy images
Computational expense	More computationally expensive	Less computationally expensive
Edge localization	Better edge localization	Less precise localization
Parameter Tuning	Standard deviation	Standard deviation

Designing an optimal 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

Canny edge detector

- The Canny edge detector is an edge detection operator that uses a multi-stage algorithm to detect a wide range of edges in images. It was developed by John F. Canny in 1986.
- This is probably the most widely used edge detector in computer vision

Canny Algorithm

- The Process of Canny edge detection algorithm can be broken down to 5 different steps:
  1. Apply Gaussian filter to smooth the image in order to remove the noise
  2. Find the intensity gradients of the image
  3. Apply non-maximum suppression to get rid of spurious response to edge detection
  4. Apply double threshold to determine potential edges
  5. Track edge by hysteresis: Finalize the detection of edges by suppressing all the other edges that are weak and not connected to strong edges.

Canny Algorithm – Step 1

- Since edge detection is susceptible to noise in the image, first step is to remove the noise in the image with a 5x5 Gaussian smoothing filter.

$\frac{1}{159}$

2	4	5	4	2
4	9	12	9	4
5	12	15	12	5
4	9	12	9	4
2	4	5	4	2

Canny Algorithm – Step 2

- The second step is to use Sobel masks to find the edge gradient strength and direction for each pixel.
  - The magnitude, or edge strength, of the gradient is then approximated using the formula: |G| = |Gx| + |Gy|

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

Gx

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

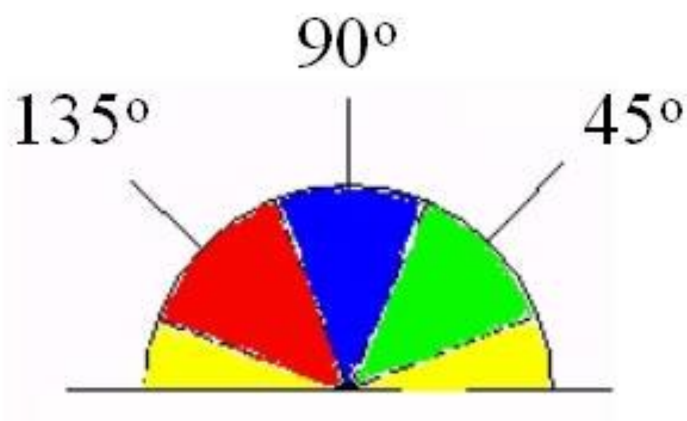
Gy

  - The direction of the edge is computed using the gradient in the x and y directions

$$\theta[i, j] = \tan^{-1}(Gy / Gx)$$

Canny Algorithm – Step 3

- Gradient Orientation
  - Reduce angle of Gradient θ[i,j] to one of the 4 sectors
  - Check the 3x3 region of each M[i,j]
    - Any edge direction falling within the **yellow range** (0 to 22.5 & 157.5 to 180 degrees) is set to 0 degrees. Any edge direction falling in the **green range** (22.5 to 67.5 degrees) is set to 45 degrees. Any edge direction falling in the **blue range** (67.5 to 112.5 degrees) is set to 90 degrees. And finally, any edge direction falling within the **red range** (112.5 to 157.5 degrees) is set to 135 degrees.

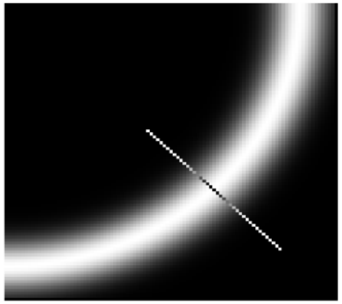


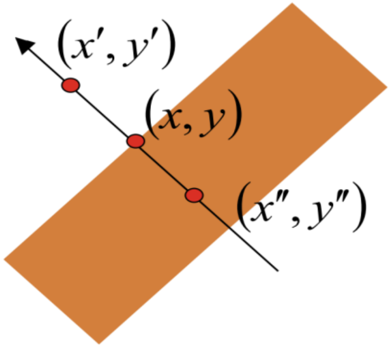
Canny Algorithm – Step 4

- Non-maximum suppression is an edge thinning technique.
- The edge extracted from the gradient value is still quite blurred.
- Non-maximum suppression can help to suppress all the gradient values to 0 except the local maximal, which indicates location with the sharpest change of intensity value.
- The algorithm for each pixel in the gradient image is:
  - Compare the edge strength of the current pixel with the edge strength of the pixel in the positive and negative gradient directions.
  - If the edge strength of the current pixel is the largest compared to the other pixels in the mask with the same direction (i.e., the pixel that is pointing in the y direction, it will be compared to the pixel above and below it in the vertical axis), the value will be preserved. Otherwise, the value will be suppressed.

Canny Algorithm – Step 4

- **Non-maximum Suppression**
  - For each pixel (i,j):
    - Find the direction d_k, which best approximates the direction
    - If M[i,j] is smaller than at least one of its two neighbors along d_k assign I[i,j]=0; otherwise assign I[i,j]=M[i,j]





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

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

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

Canny Algorithm – Step 4

- **Non-maximum Suppression: an Example**

0	0	0	0	1	1	1	3
3	0	0	1	2	1	3	1
0	0	2	1	2	1	1	0
0	1	3	2	1	1	0	0
0	3	2	1	0	0	1	3
2	3	2	0	0	1	0	1
2	3	2	0	1	0	2	1

0	0	0	0	0	0	0	3
3	0	0	0	2	1	3	0
0	0	2	1	2	0	0	0
0	0	3	0	0	0	0	0
0	3	2	0	0	0	0	0
0	3	0	0	0	1	0	1
0	3	0	0	1	0	2	0

Canny Algorithm – Step 4

- Non-maximum Suppression: an Example

Before After

Canny Algorithm – Step 5

- **Hysteresis Thresholding**
  - The image output by NONMAX- SUPPRESSION I[i,j] still contains the local maxima created by noise. How do we get rid of these?
  - Reduce number of false edges by applying a threshold T
    - All values below T are changed to 0
    - Selecting a good values for T is difficult
    - Some false edges will remain if T is too low
    - Some edges will disappear if T is too high
    - Some edges will disappear due to softening of the edge contrast by shadows
  - 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
      - Consider its neighbors iteratively then declare it as "edge pixel" if it is connected to an 'strong edge pixel' directly or via pixels between Low and High

Canny Algorithm – Step 5

- **Hysteresis Thresholding:**
  - Double Thresholding
    - Two threshold values, T_L and T_H are applied to I[i,j].
    - Here $T_L < T_H$
    - Two images in the output
    - The image from T_H contains fewer edges but has gaps in the contours
    - The image from T_L has many false edges
    - Combine the results from T_L and T_H
    - Link the edges of T_H into contours until we reach a gap
    - link the edge from T_H with edge pixels from a T_L contour until a T_H edge is found again

Canny Algorithm – Step 5

- **Hysteresis Thresholding**

$T_H=2$

0	0	0	0	0	0	0	3
0	0	0	0	2		3	0
0	0	2		2	0	0	0
0	0	3	0	0	0	0	0
0	3	2	0	0	0	0	0
0	3	0	0	0	0	0	0
0	3	0	0	0	0	2	0

gaps filled from T_1

$T_L=1$

0	0	0	0	0	0	0	3
0	0	0	0	2	1	3	0
0	0	2	1	2	0	0	0
0	0	3	0	0	0	0	0
0	3	2	0	0	0	0	0
0	3	0	0	0	1	0	1
0	3	0	0	1	0	2	0

- A T_H contour has pixels along the green arrows
- Linking: search in a 3x3 of each pixel and connect the
- pixel at the center with the one having greater value
- Search in the direction of the edge (direction of Gradient)

Hysteresis thresholding example

original image

high threshold
(strong edges)

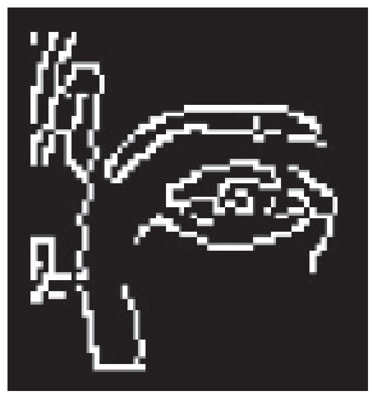
low threshold
(weak edges)

hysteresis threshold

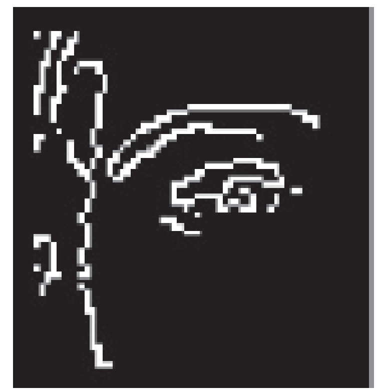
Canny Algorithm – Step 5

- Hysteresis Thresholding

Gaussian smoothing



(a) Hysteresis thresholding, upper level = 40, lower level = 10

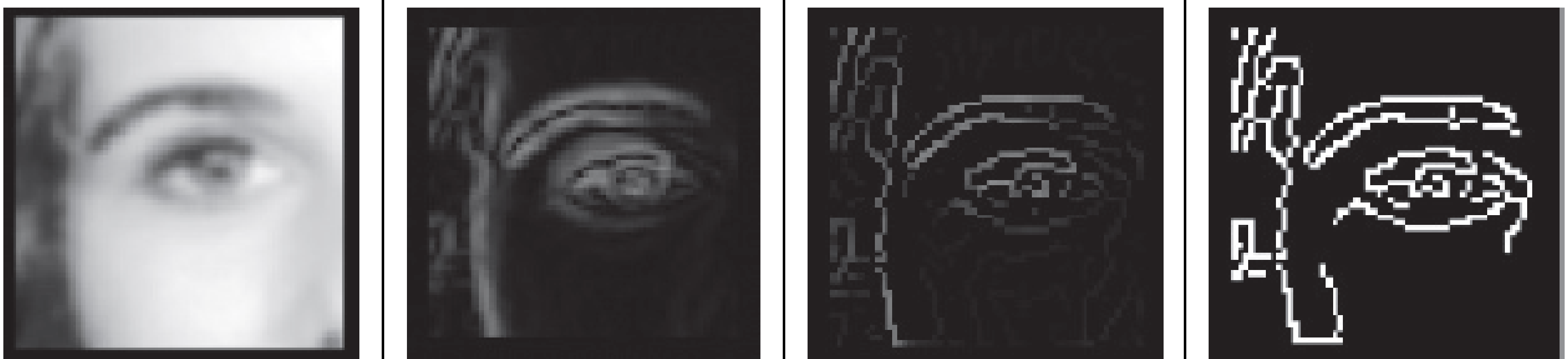


(b) Uniform thresholding, level = 40

(c) Uniform thresholding, level = 10

Canny Algorithm

- Stages in Canny edge detection - Example



(a) Gaussian smoothing | (b) Sobel edge detection | (c) Non-maximum suppression | (d) Hysteresis thresholding

Effect of Gaussian Kernel (smoothing)

original

Canny with $\sigma = 1$

Canny with $\sigma = 2$

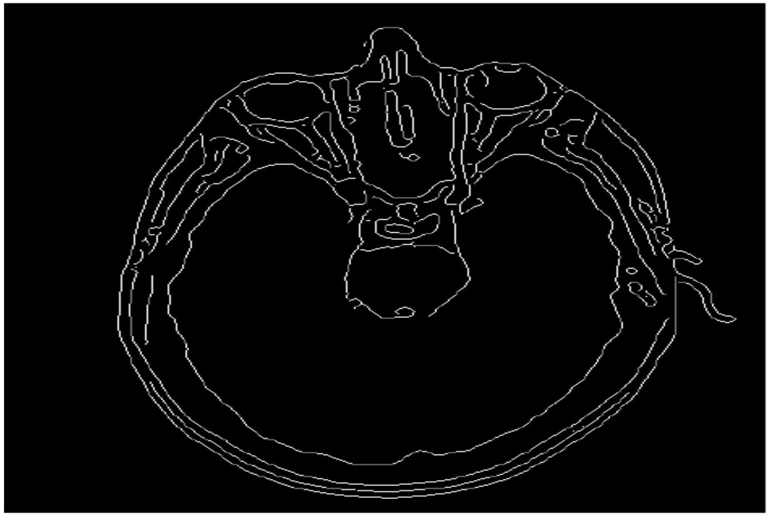
The choice of σ depends on desired behavior

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

Canny Edge Detection Summery

- The performance of the Canny algorithm depends heavily on the adjustable parameters, , which is δ and the threshold values, ‘T1’ and ‘T2’.
  - The bigger the value for δ, the larger the size of the Gaussian filter becomes. This implies more blurring, necessary for noisy images, as well as detecting larger edges.
  - However, the larger the scale of the Gaussian, the less accurate is the localization of the edge.
  - The user can tailor the algorithm by adjusting these parameters to adapt to different environments.
  - Canny’s edge detection algorithm is computationally more expensive compared to Sobel, Prewitt and Robert’s operator. However, the Canny’s edge detection algorithm performs better than all these operators under almost all scenarios

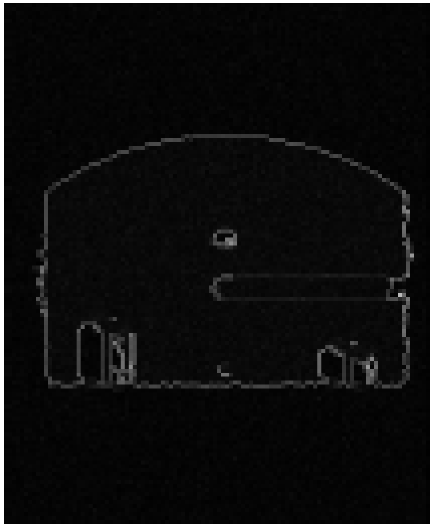
Canny Edge Detection - Examples



Edge Detection Comparison Example

(a)

(b)

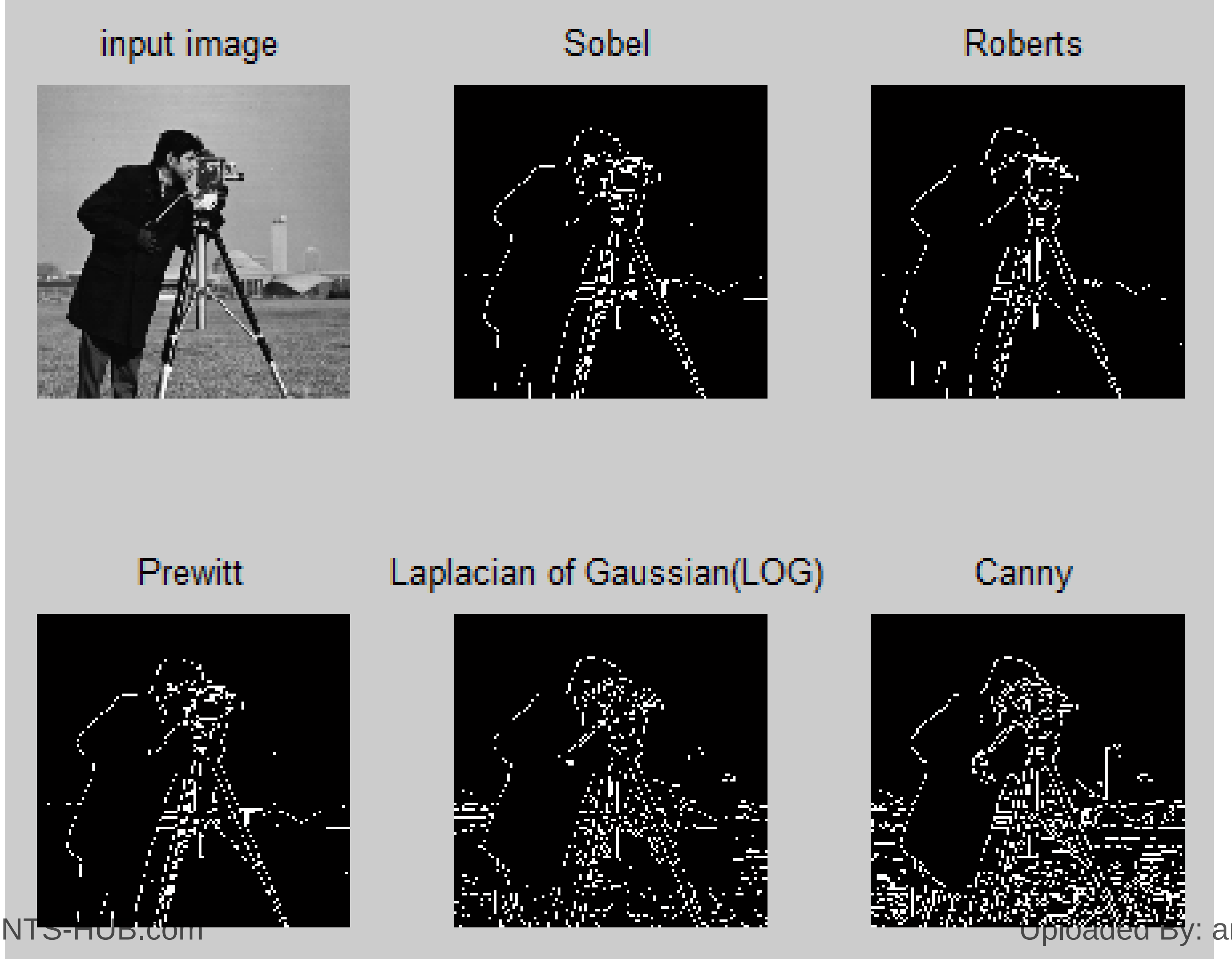


(c)

(d)

- (a) Original Image with Noise
- (b) Sobel
- (c) Robert
- (d) Canny

Edge Detection Comparison Example



Edge Detection Other Approaches

- SOFT COMPUTING APPROACHES
  - *Fuzzy based Approach*
  - *Genetic Algorithm Approach*
  - *Neural Network Approach – Deep Learning*

- Soft computing approaches, are applied on a real life example image of nature scene

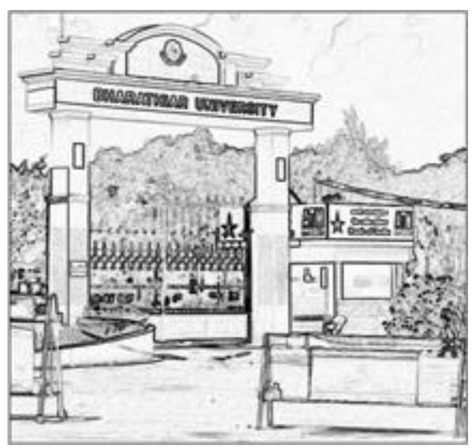
Edge Detection Other Approaches

Original

Roberts

Sobel

Fuzzy

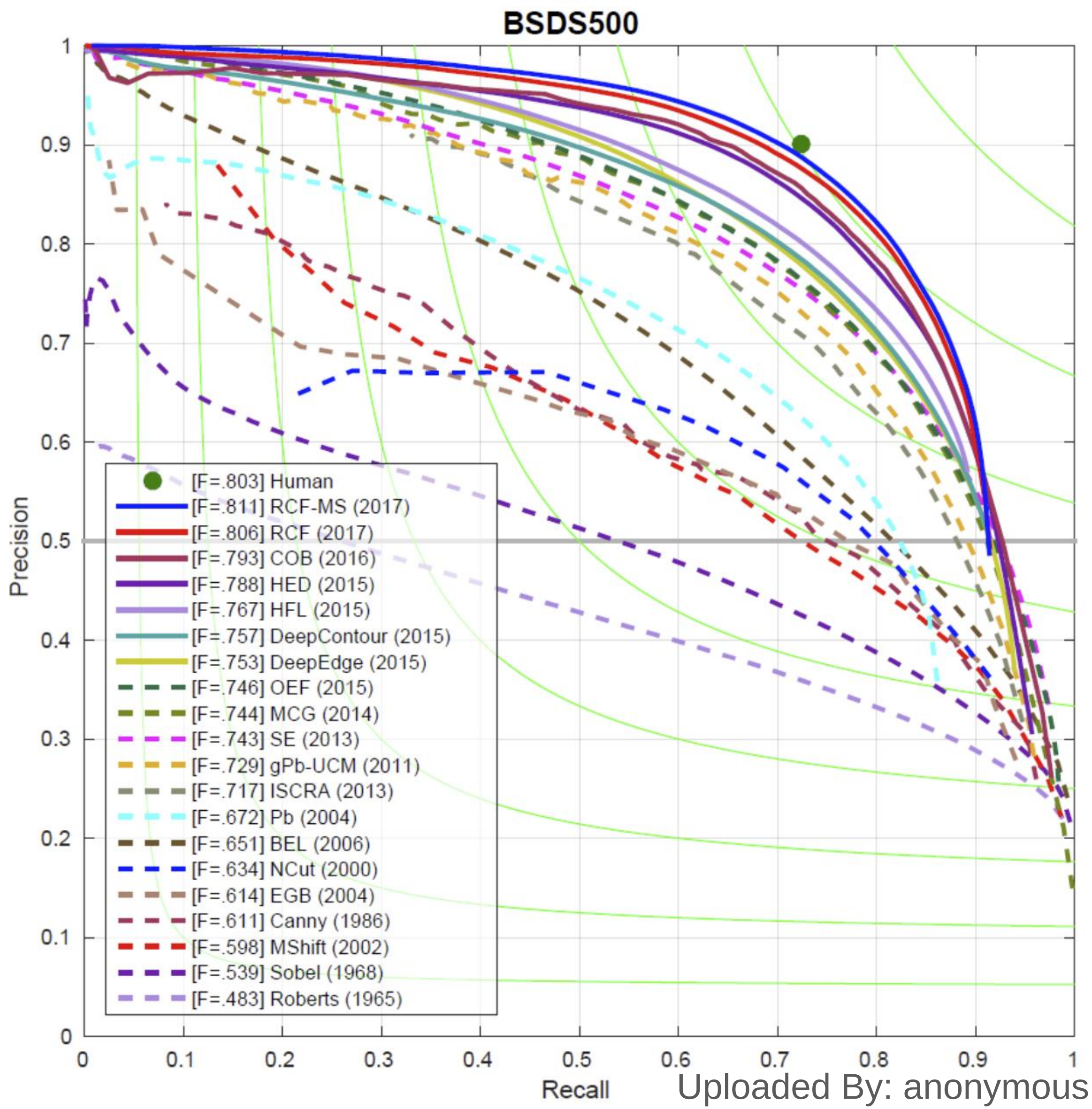


Genetics

Neural Network

Edge Detection with Deep Learning

- We will revisit edge detection after Deep Learning tutorial lectures



Acknowledgement

- The material in these slides are based on:
  - Digital Image Processing: Rafael C. Gonzalez, and Richard
  - Forsythe and Ponce: Computer Vision: A Modern Approach
  - Rick Szeliski's book: Computer Vision: Algorithms and Applications
  - cs131@ Stanford University
  - cs131n@ Stanford University
  - CS198-126@ University of California, Berkely
  - CAP5415@ University of Central Florida
  - CSW182 @ University of California, Berkely
  - 11-785@ Carnegie Mellon University
  - CSCI1430@ Brown University
  - Computer Vision@ Bonn University
  - ICS 505@ KFUPM
  - Digital Image Processing@ University of Jordan