CSE4227 Digital Image Processing

Chapter 10 – **Image Segmentation (Part I)**

Dr. Kazi A Kalpoma

Professor, Department of CSE

Ahsanullah University of Science & Technology (AUST)

Contact: kalpoma@aust.edu

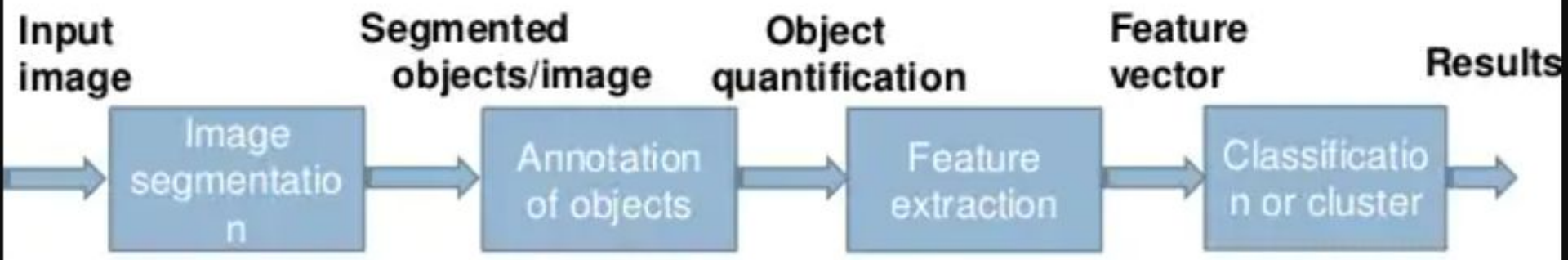
Today's Contents

- **Image Segmentation**
  - **Definition**
  - **Goal**
  - **Application**
- **Types of Segmentation Algorithms**
- Detection Of Discontinuities
  - Points
  - Lines
- Concept of Edge Detection
- Canny Edge Detection Algorithm

•Chapter 10 from R.C. Gonzalez and R.E. Woods, Digital Image Processing (3rd Edition), Prentice Hall, 2008 [**Section 10.1, 10.2 (excluding 10.2.7)**]

What is Image Segmentation ?

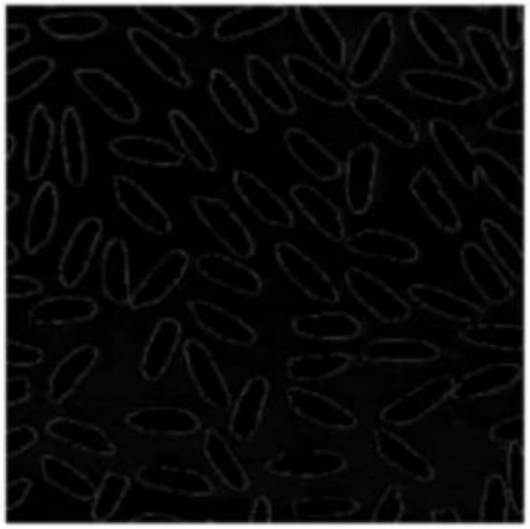
- Image segmentation is an aspect of image processing.
- *Image segmentation is a computer vision process.*
- *Image segmentation is the first step in image analysis.*



A typical image analysis pipeline.

Simple example:

segmentation of rice grains



Original image

Segmented (binary) image

Each pixel is assigned a label:

- 0 = not rice grain pixel
- 1 = rice grain pixel

What does Image Segmentation do?

- Divide the image into different regions.
- Separate objects from background and give them individual ID numbers (labels).
- Purpose is to partition an image into meaningful regions with respect to a particular application.

For example, it allows us to:

- Count the number of objects of a certain type.
- Measure geometric properties (e.g., area, perimeter) of objects in the image.
- Study properties of an individual object (intensity, texture, etc.)

Principle Approaches of Segmentation

Segmentation Algorithms generally are based on one of the two basis properties of intensity values:

- **Similarity**
  - Partitioning an image into regions that are similar according to a set of predefines criteria – (thresholding, region growing, region splitting and merging)
- **Discontinuity**
  - Partitioning an image based on sharp changes in intensity (such as edges in an image).

Types of Segmentation Algorithms

❑Similarity

- Thresholding – based on pixel intensities
- Region based – grouping similar pixels
- Match based – comparison to a given template

❑Discontinuity

- Edge based – detection of edges that separate regions from each other
- Watershed – find regions corresponding to local minima in intensity

Detection Of Discontinuities

- There are three basic types of gray-level discontinuities:

  * points * lines * edges

- We are interested in the behavior of the derivatives in these discontinuities.

- The common way of achieving this detection is to run a mask through the image.

Point Detection

−1	−1	−1
−1	8	−1
−1	−1	−1

- a point has been detected at the location on which the mark is centered if

$$|R| \geq T$$

- where
  - T is a nonnegative threshold
  - R is the sum of products of the coefficients with the gray levels contained in the region encompassed by the mark.

$$R = \sum_{k=1}^{9} w_k z_k$$

w_1	w_2	w_3
w_4	w_5	w_6
w_7	w_8	w_9

Detection of Isolated Points

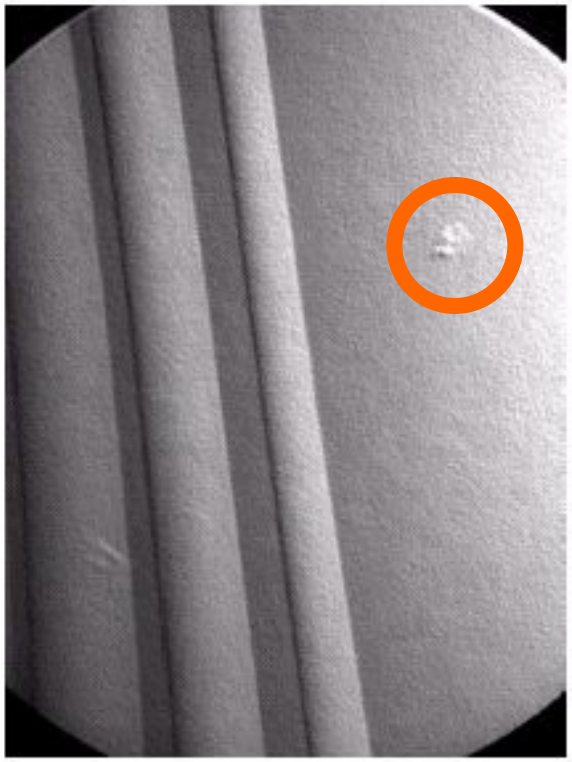
- Point detection can be achieved simply using the Laplacian mask.

-1	-1	-1
-1	8	-1
-1	-1	-1

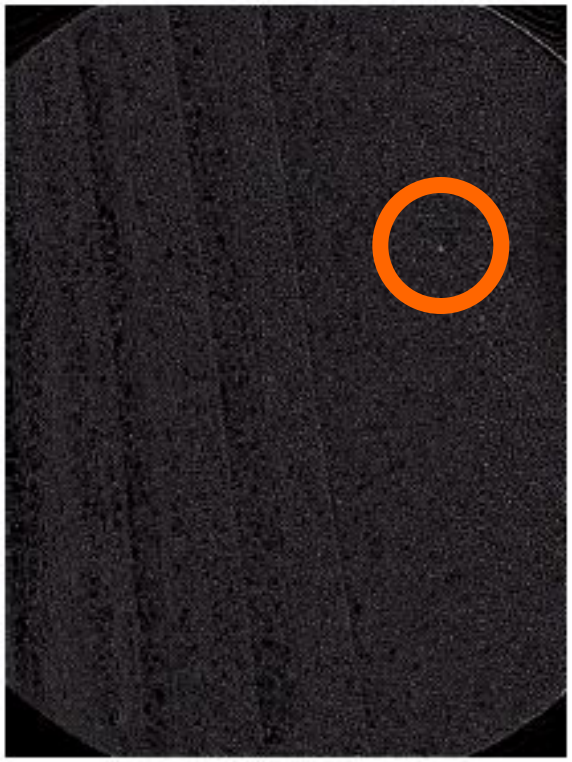
$$g(x,y) = \begin{cases} 1 & \text{if } |R(x,y)| \geq T \\ 0 & \text{otherwise} \end{cases} \qquad R = \sum_{k=1}^{9} w_k z_k$$

Points are detected at those pixels in the subsequent filtered image that are above a set threshold

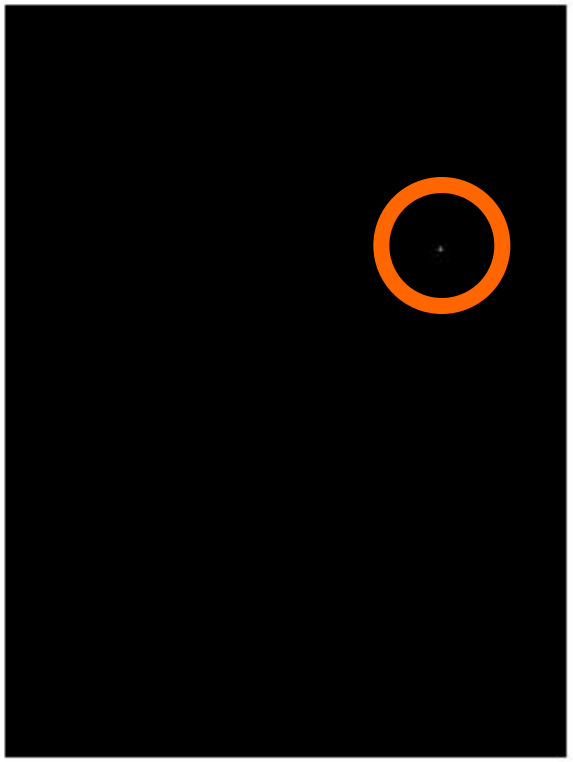
Isolated Point Detection (example)



X-ray image of
a turbine blade



Result of point
detection



Result of
thresholding

Line Detection

- The masks below will extract lines that are one pixel thick and running in a particular direction

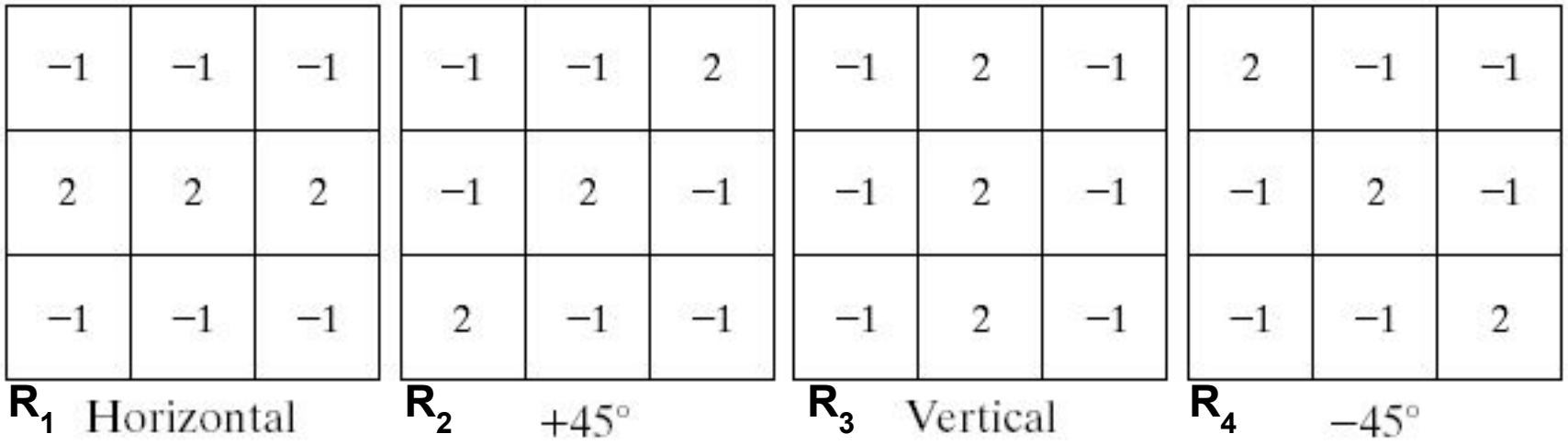
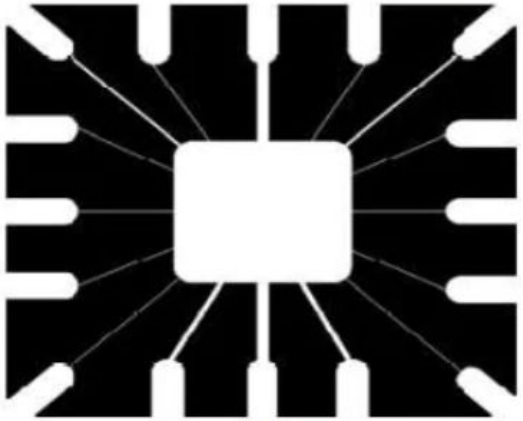


FIGURE 10.6 Line detection masks. Angles are with respect to the axis system in Fig. 2.18(b).

- Note: preferred direction of each mask is weighted with a larger coefficient (i.e.,2) than other possible directions.

Example



(a)

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

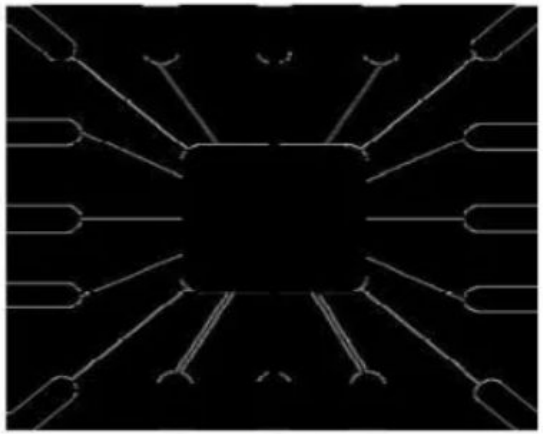
(b)

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

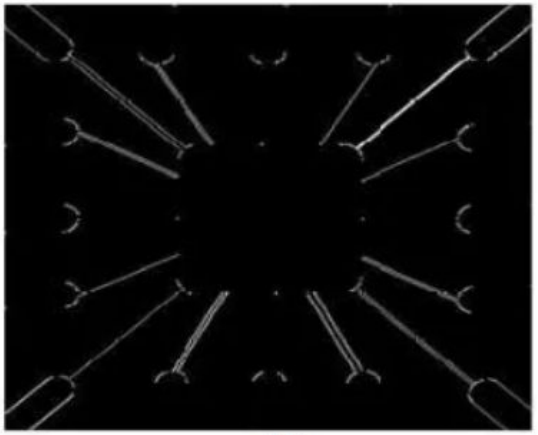
(c)

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

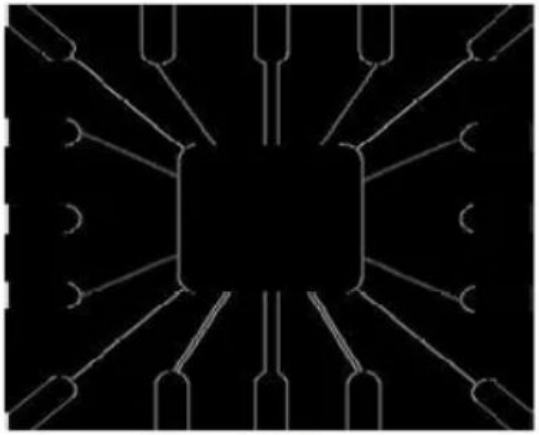
(d)



(e)



(f)



(g)

Line detection. (a) Image of a wire-bond mask; (b) Horizontal line detector mask; (c) +45° line detector mask; (d) Vertical line detector mask; (e) Result of processing with the horizontal line detector mask; (f) Result of processing with the +45° line detector mask; (g) Result of processing with the vertical line detector mask.

Line Detection

- Apply every masks on the image
- let R_1, R_2, R_3, R_4 denotes the response of the horizontal, +45 degree, vertical and -45 degree masks, respectively.
- if, at a certain point in the image

$|R_i| > |R_j|$, for all $j \neq i$,

- that point is said to be more likely associated with a line in the direction of mask i.

Line Detection

- Alternatively, if we are interested in detecting all lines in an image in the direction defined by a given mask, we simply run the mask through the image and threshold the absolute value of the result.
- The points that are left are the strongest responses, which, for lines one pixel thick, correspond closest to the direction defined by the mask.

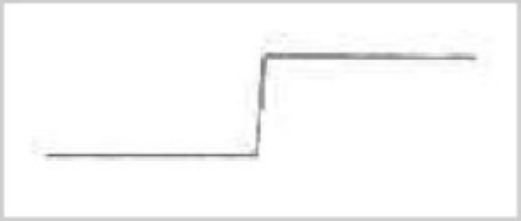
Edges

- Abrupt change in the intensity of pixels.
- Discontinuity in image brightness or contrast.
- Usually edges occur on the boundary of two regions .

❑ Edge is the boundary between two homogeneous regions.

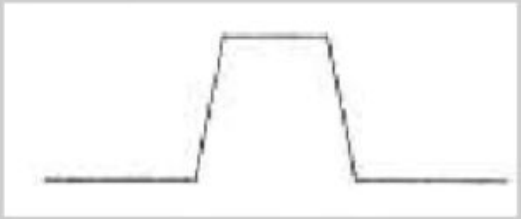
Edge Types

- Step Edge

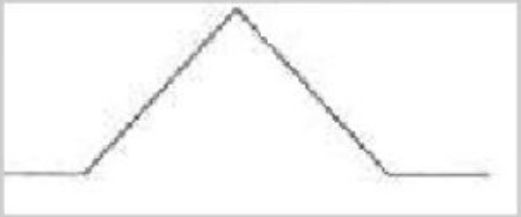


- Ramp Edge

- Ridge



- Roof



Edges & Derivatives

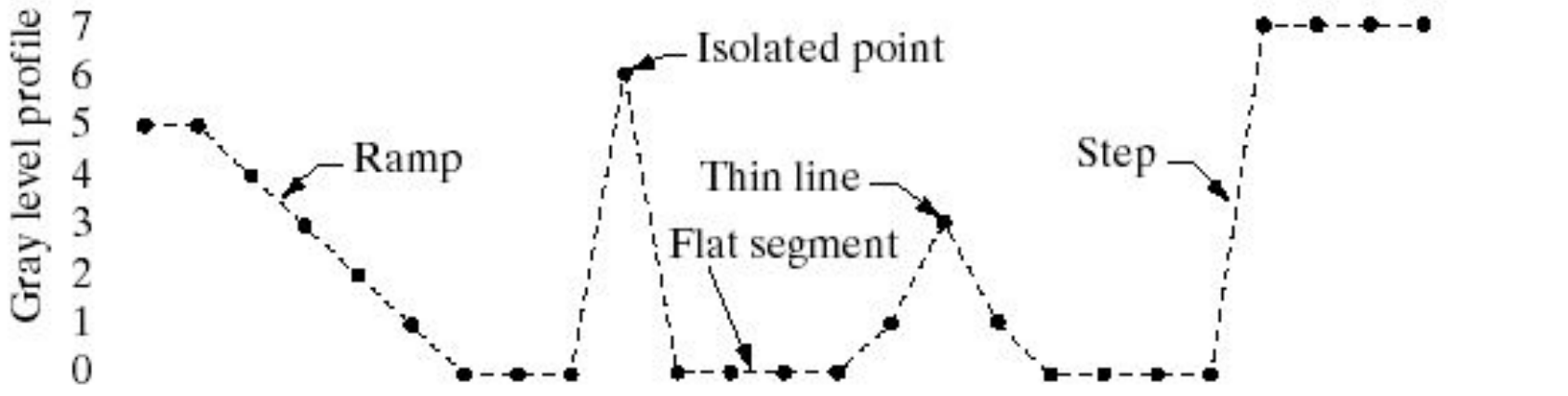
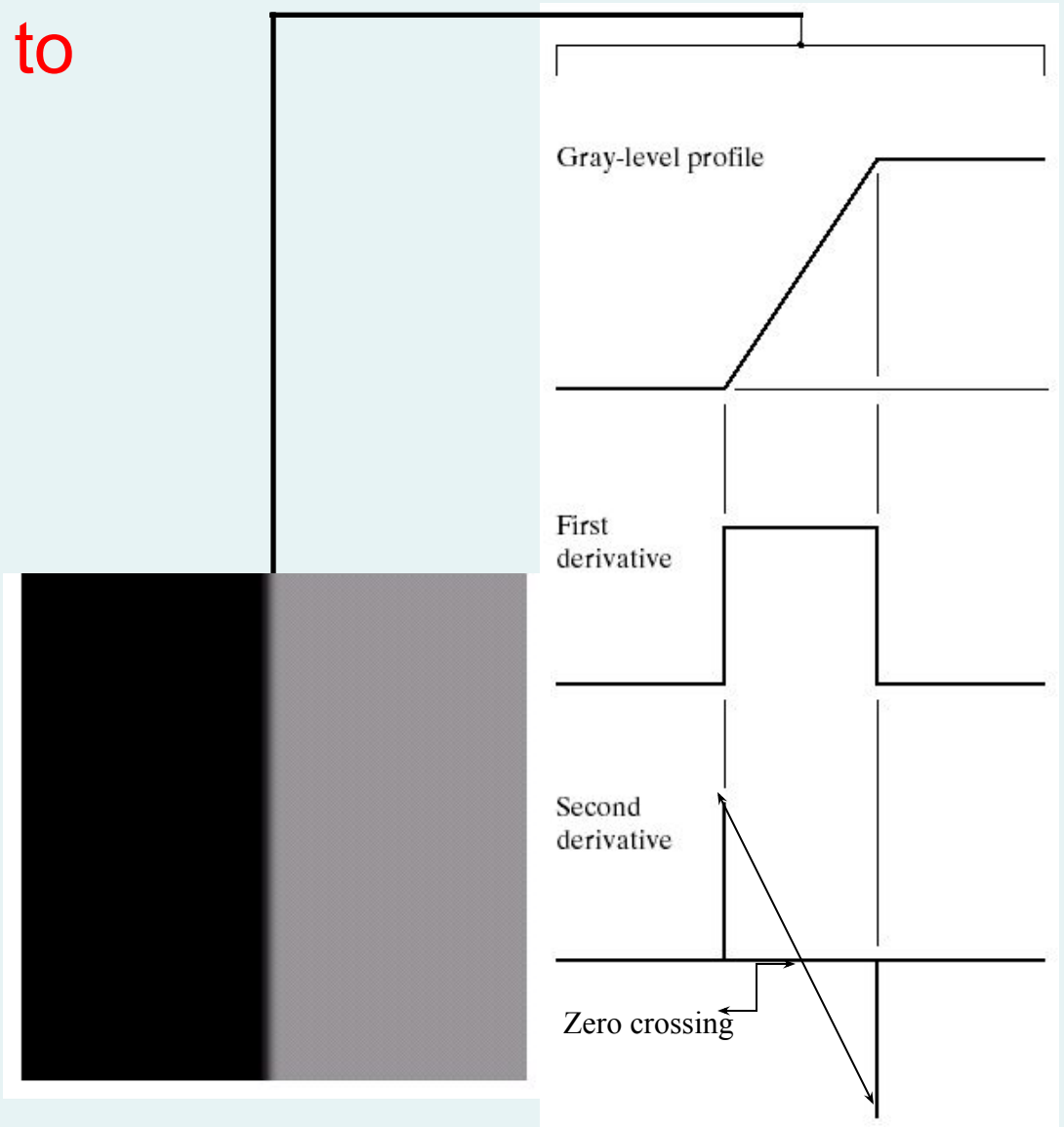


Image strip	5	5	4	3	2	1	0	0	0	6	0	0	0	0	1	3	1	0	0	0	0	7	7	7	7	•	•
1st Derivative		-1	-1	-1	-1	-1	0	0	6	-6	0	0	0	1	2	-2	-1	0	0	0	7	0	0	0			
2nd Derivative		-1	0	0	0	0	1	0	6	-12	6	0	0	1	1	-4	1	1	0	0	7	-7	0	0			

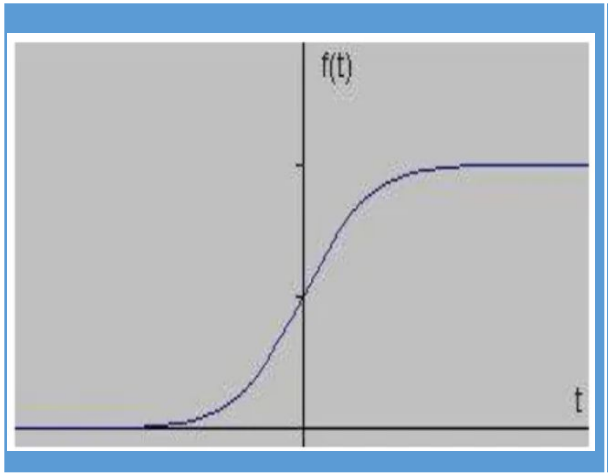
- First order derivative produce thick edge
- 2nd order derivative produce much finer one and strong response at isolated point

Edges & Derivatives

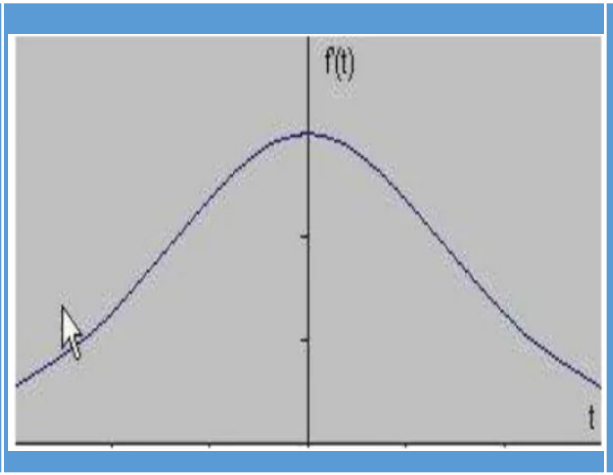
- Derivatives are used to find discontinuities
- 1st derivative tells us where an edge is
- 2nd derivative can be used to show edge direction



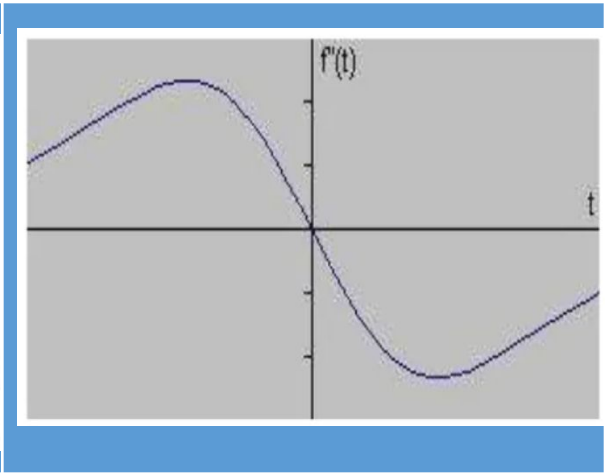
Characteristics of First and Second Order Derivatives



Example Signal



1st Derivative of Signal

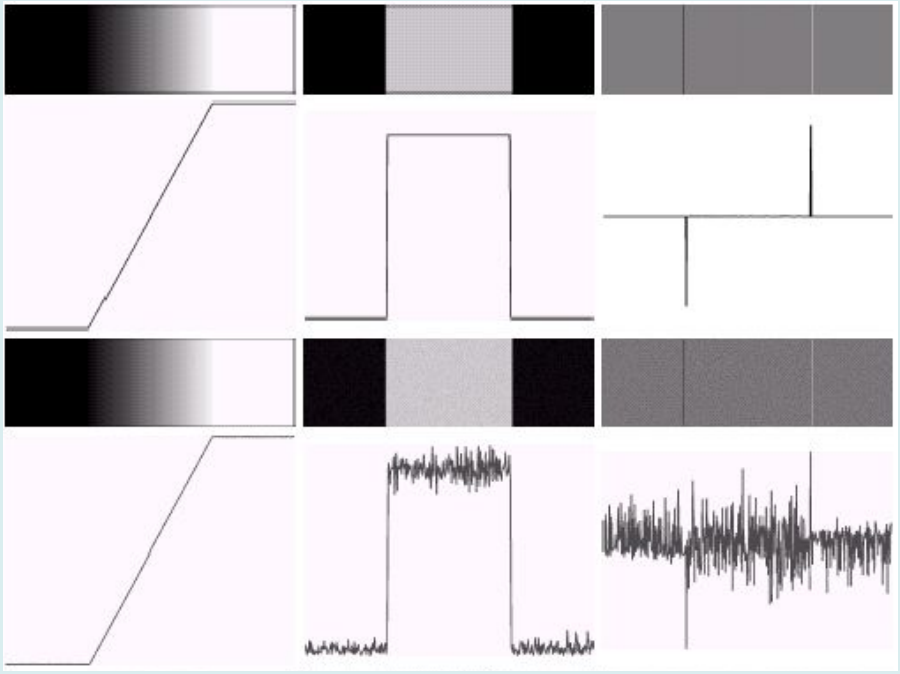


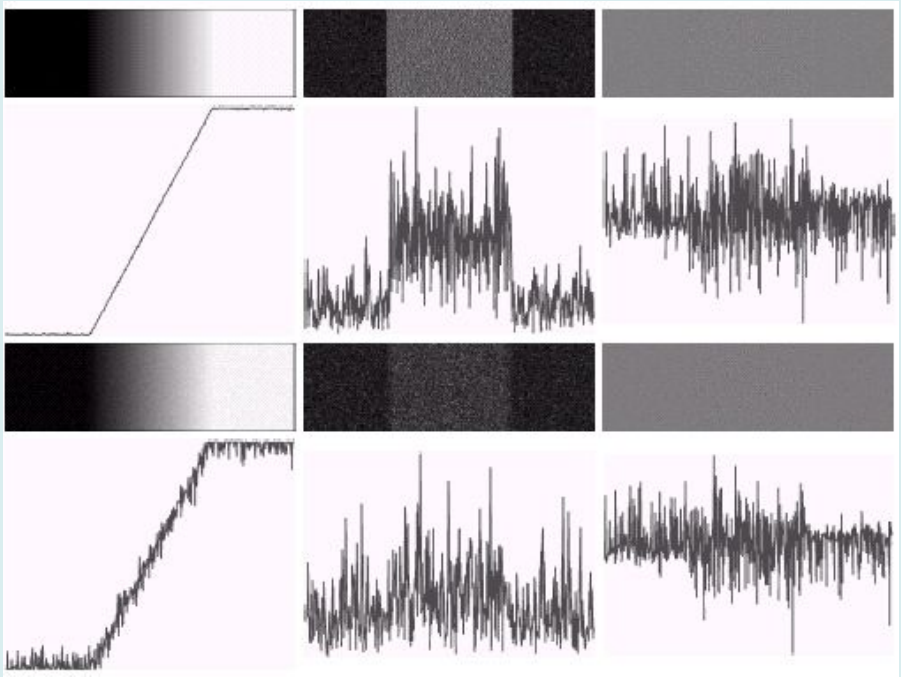
2nd Derivative of Signal

- 1ST DERIVATIVE SHOWS A MAXIMUM AT THE CENTER OF THE EDGE SIGNAL.
- 2ND DERIVATIVE PASSES THROUGH ZERO (CHANGES ITS SIGN) AT THE CENTER OF THE EDGE SIGNAL.

Derivatives & Noise

- Derivative based edge detectors are extremely sensitive to noise
- We need to keep this in mind

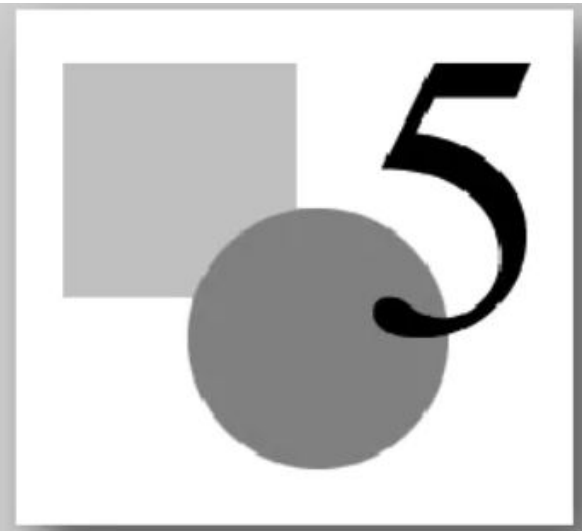




Edge Detection

- Process of identifying edges in an image to be used as a fundamental asset in image analysis.
- Locating areas with strong intensity contrasts.

Edge Detection - Identifying sudden change in image intensity.



Edge Detection Usage

- Reduce unnecessary information in the image while preserving the structure of the image.
- Extract important features of an image
  - Corners
  - Lines
  - Curves
- Recognize objects, boundaries, segmentation.
- Part of computer vision and recognition.

Methods of Edge Detection

- The majority of different edge detection methods may be grouped into two categories:

- **Gradient based Edge Detection**:

  The gradient method detects the edges by looking for the **maximum and minimum** in the first derivative of the image.

- **Laplacian based Edge Detection:**

  The Laplacian method searches for **zero crossings** in the second derivative of the image to find edges. An edge has the one-dimensional shape of a ramp and calculating the derivative of the image can highlight its location.

Edge Detection Algorithms

- Based on those category we can group algorithm like below
  - **Gradient based**
    - Sobel Operator
    - Robert's cross operator
    - Prewitt's operator
  - **Laplacian based**
    - Laplacian of Gaussian
  - **Hybrid (both Gradient and Laplacian)**
    - Canny Edge Detection Algorithm

Three main steps in Edge Detection

1. Filtering (Smoothing)

   In this stage image is pass through a filter to remove the noise.

2. Differentiation (Edge sharpening using derivatives)

   this stage highlights the location in the image where intensity changes i.e. detects discontinuities.

3. Detection (Thresholding)

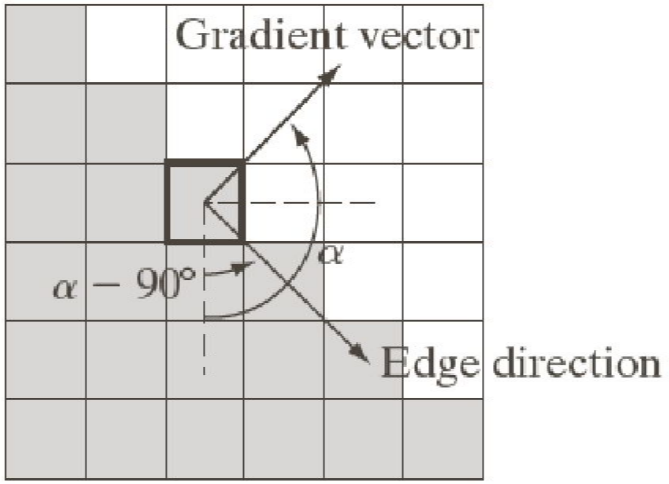
   this stage take decision on the edge pixel i.e. where the changes are significant.

4. Localization

   determine the exact location of an edge.

Gradient based Edge Detection

- $I = f(x, y)$
- $\vec{\nabla} f = \begin{bmatrix} G_x \\ G_y \end{bmatrix} = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix}$



- $\nabla f = mag(\vec{\nabla} f) = \sqrt[2]{{G_x}^2 + {G_y}^2} \approx |G_x| + |G_y|$
- Direction of $\vec{\nabla} f$, $\alpha(x, y) = \tan^{-1}\left(\frac{G_y}{G_x}\right)$

❖ The direction of an edge θ is orthogonal to the direction of a gradient vector α

Gradient – Prewitt Operator

- 3X3 Convolution Mask
- Convolution Mask
  - G_x

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

G_y

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

- The differences are calculated at the center pixel of the mask.

Gradient – Sobel Operator

- 3X3 Convolution Mask
- Convolution Mask
  - G_x

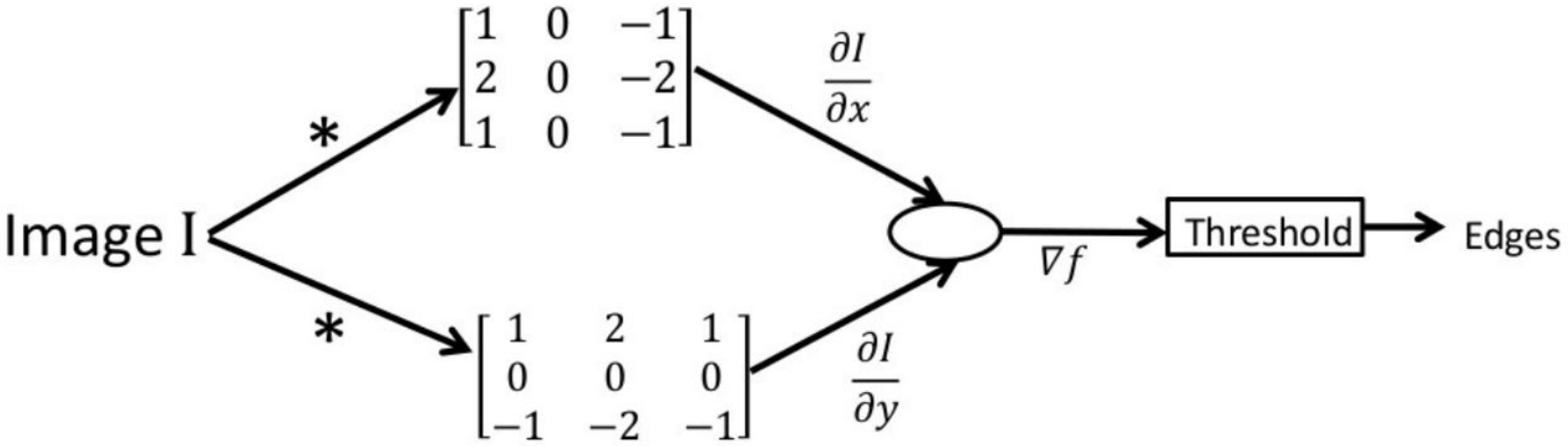
-1	0	1
-2	0	2
-1	0	1

G_y

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

- The differences are calculated at the center pixel of the mask.

Basic Edge Detection by Sobel



Sobel Edge Detector

Gradient based Edge Detection cont.

- Simple to implement
- Capable of detecting edges and their direction
- Sensitive to noise
- Not accurate in locating edges

Edge Detection Example

Original Image

Horizontal Gradient Component

Vertical Gradient Component

Combined Edge Image

Edge Detection Example

Edge Detection Problems

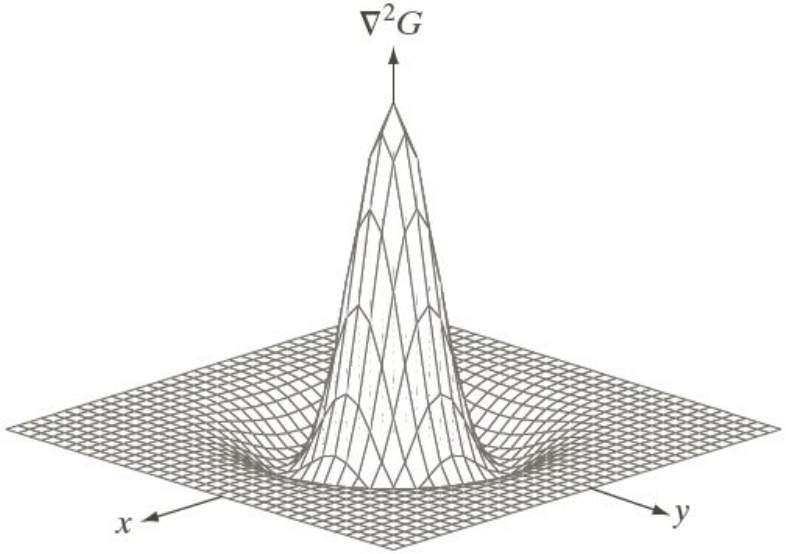
- Often, problems arise in edge detection in that there are too much details
- For example, the brickwork in the previous example
- One way to overcome this is to smooth images prior to edge detection

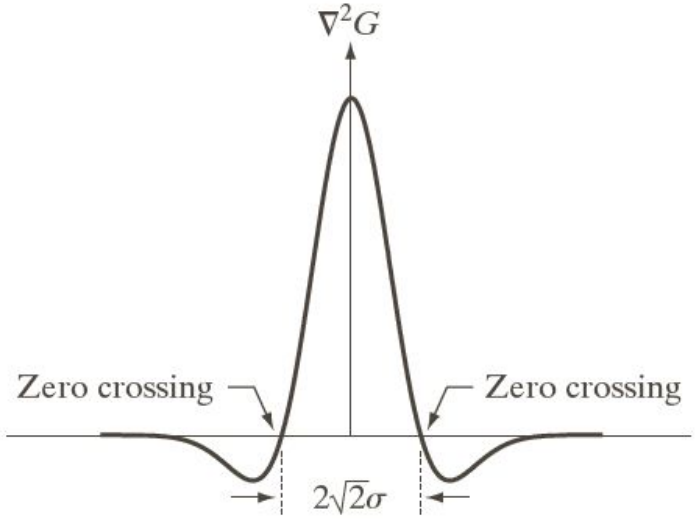
Laplacian of Gaussian – LOG

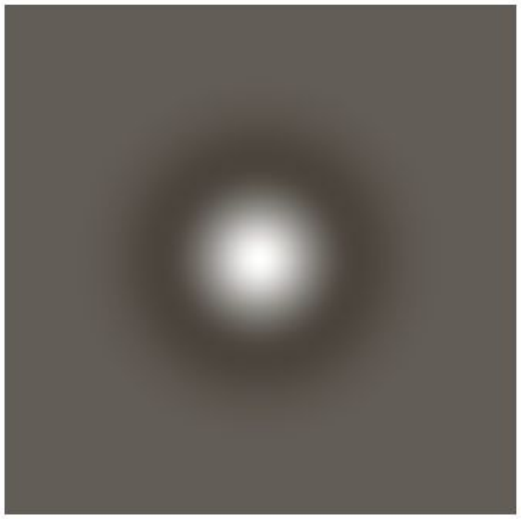
- Steps:
  - Smoothing: Gaussian filter
  - Enhance edges: Laplacian operator

$$g(x,y) = \nabla^2 \left[G(x,y) \star f(x,y) \right]$$

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

a b
c d

FIGURE 10.21 (a) Three-dimensional 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 mask approximation to the shape in (a). The negative of this mask would be used in practice.

Laplacian Of Gaussain

Laplacian of Gaussian - LOG

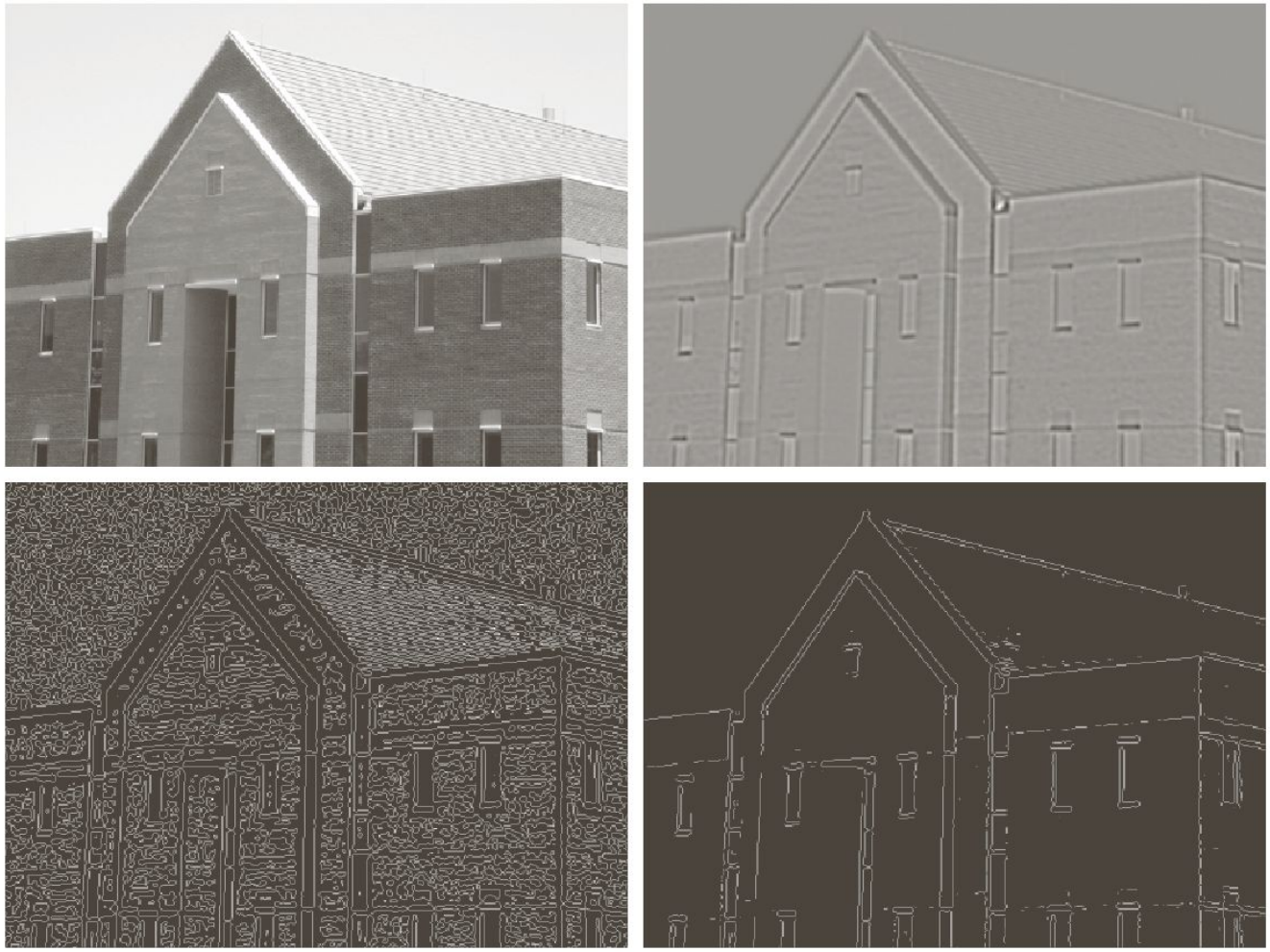
- Computationally cheaper to implement since we can combine the two filters into one filter but it.
- Doesn't provide information about the direction of the edge.
- Probability of false and missing edges remain.
- Localization is better than Gradient Operators

Laplacian of Gaussian – LOG (Marr Hildreth Algorithm)

- Steps:
  - Smoothing: Gaussian filter
  - Enhance edges: Laplacian operator
  - Zero crossings denote the edge location
  - Use linear interpolation to determine the sub-pixel location of the edge

$$g(x,y) = \nabla^2 \left[G(x,y) \star f(x,y) \right]$$

Marr-Hildreth Algorithm



a b
c d

FIGURE 10.22
(a) Original image of size 834 × 1114 pixels, with intensity values scaled to the range [0, 1]. (b) Results of Steps 1 and 2 of the Marr-Hildreth algorithm using $\sigma = 4$ and $n = 25$. (c) Zero crossings of (b) using a threshold of 0 (note the closed-loop edges). (d) Zero crossings found using a threshold equal to 4% of the maximum value of the image in (b). Note the thin edges.

The Canny Edge Detector

- Three main criteria
  - Good Detection : The ability to locate and mark all real edges
  - Good Localisation : Minimal distance between the detected edge and real edge
  - Clear Response : Only one response per edge

The Canny Edge Detector

The Canny Edge Detector

The algorithm runs in 5 separate steps :

1. Smoothing: Blurring of the image to remove noise.

2. Finding gradients: The edges should be marked where the gradients of the image has large magnitudes.

3. Non-maximum suppression: Only local maxima should be marked as edges.

4. Double thresholding: Potential edges are determined by thresholding.

5. Edge tracking by hysteresis: Final edges are determined by suppressing all edges that are not connected to a very certain (strong) edge.

The Canny Edge Detector: Algorithm (step 1: smoothing/noise reduction)

Let $f(x, y)$ denote the input image and $G(x, y)$ denote the Gaussian function:

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

We form a smoothed image, $f_s(x, y)$ by convolving G and f:

$$f_s(x, y) = G(x, y) \bigstar f(x, y)$$

Result after step 1

Greyscale

$f(x,y)$

Blurred

$f_s(x,y)$

The Canny Edge Detector: Algorithm (step 2: finding gradient operator)

- Compute the derivative of smoothed image $f_s(x,y)$
- Calculate the Gradient Magnitude and Direction.
- Any of the filter mask pairs can be use to get the derivatives.

- Prewitt edge operator

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

Horizontal

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

Vertical

- Sobel edge operator

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

Horizontal

g_y

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

Vertical

g_x

$$M(x,y) = \sqrt{g_x^{\,2} + g_y^{\,2}}$$

and

$$\alpha(x,y) = \arctan(g_y / g_x)$$

where $g_x = \partial f_s / \partial x$ and $g_y = \partial f_s / \partial y$

Result after step 2

Blurred
$f_s(x,y)$

Gradient Magnitude
$M(x,y)$

The Canny Edge Detector: Algorithm (step 3: Non-Max Suppression)

The gradient $M(x, y)$ typically contains wide ridge around local maxima. Next step is to thin those ridges.

Nonmaxima suppression:

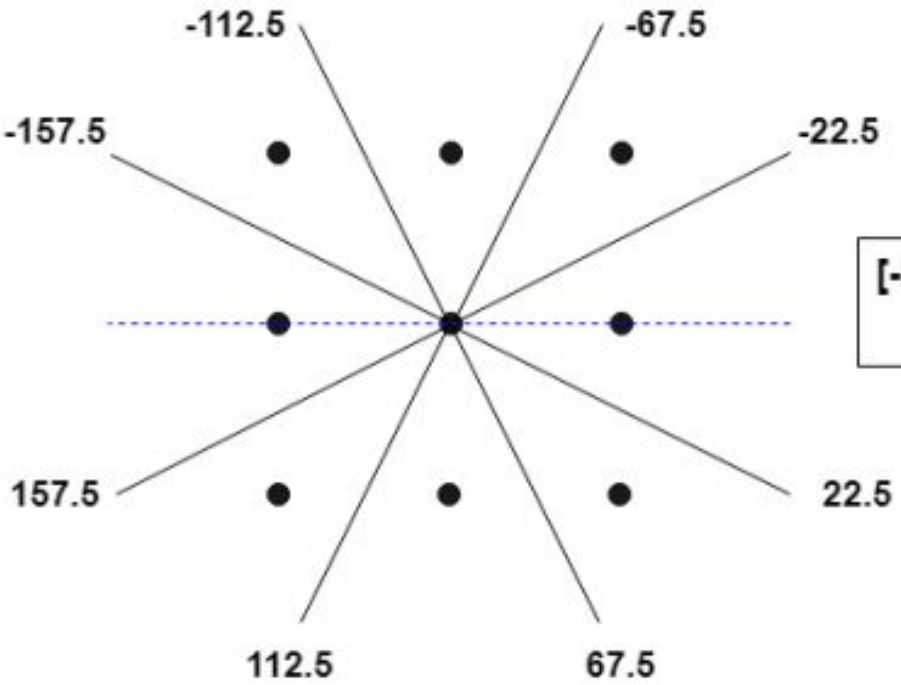
- Perform Non Maximum Suppression.
- This is an edge thinning technique.
- In this, for each pixel, need to check if it is a local maximum in its neighborhood in the direction of gradient or not.
- If it is a local maximum it is retained as an edge pixel, otherwise suppressed.

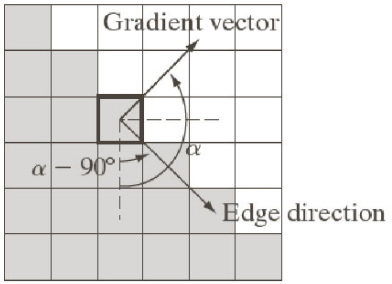
Let d_1, d_2, d_3, and d_4 denote the four basic edge directions for a 3×3 region: horizontal, -45°, vertical, $+45^\circ$, respectively.

1. Find the direction d_k that is closest to $\alpha(x, y)$.
2. If the value of $M(x, y)$ is less than at least one of its two neighbors along d_k, let $g_N(x, y) = 0$ (suppression); otherwise, let $g_N(x, y) = M(x, y)$

The Canny Edge Detector: Algorithm (step 3: Non-Max Suppression)……

- Fro each pixel, the neighboring pixels are located in horizontal, vertical, and diagonal directions (0°, 45°, 90°, and 135°).
- Thus we need to round off the gradient direction at every pixel to one of these directions as shown below.

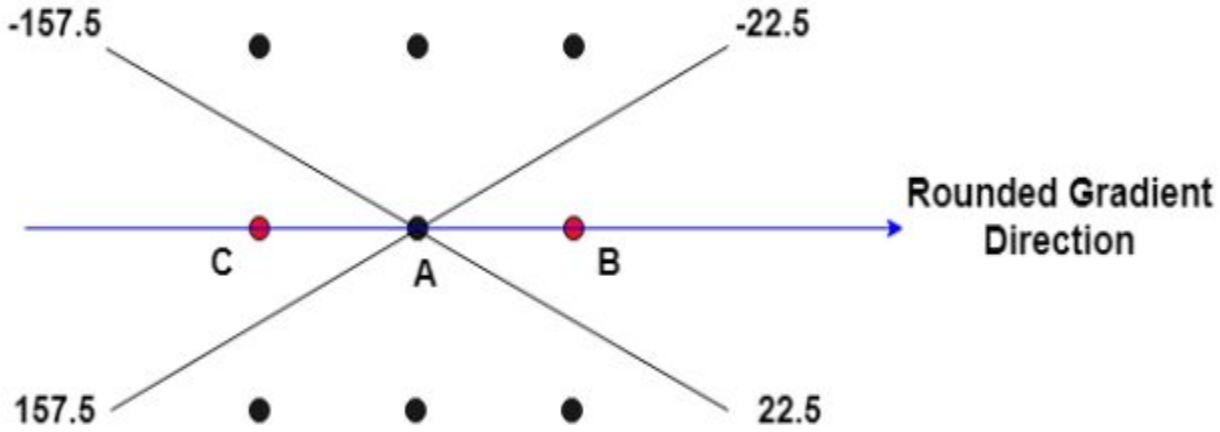




For example

[-22.5°, 22.5°] or [-157.5°, 157.5°] maps to 0°

The Canny Edge Detector: Algorithm (step 3: Non-Max Suppression)……



Example:

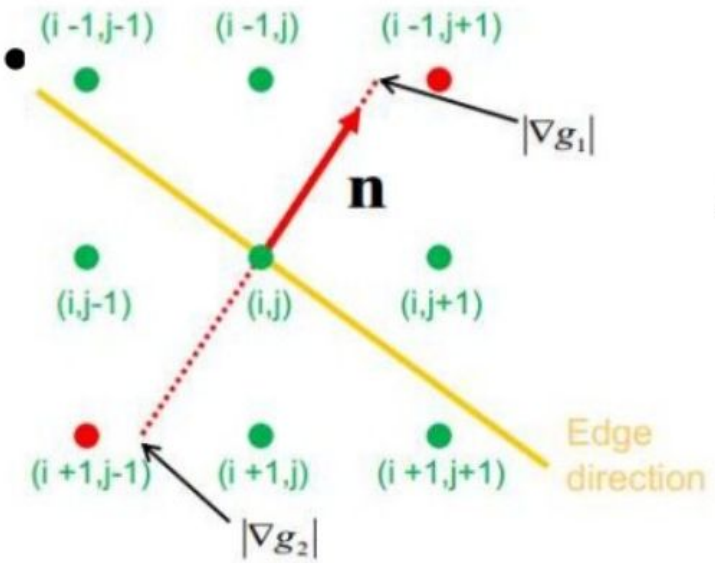
- Suppose for a pixel 'A', the gradient direction comes out to be 17 degrees.
- Since 17 is nearer to 0, we will round it to 0 degrees.
- Then we select neighboring pixels in the rounded gradient direction (See B and C in below figure).
- If the magnitude value M(x,y) of A is greater than that of B and C, it is retained as an edge pixel otherwise suppressed.

The Canny Edge Detector: Algorithm (step 3: Non-Max Suppression)

Canny Edge Detector

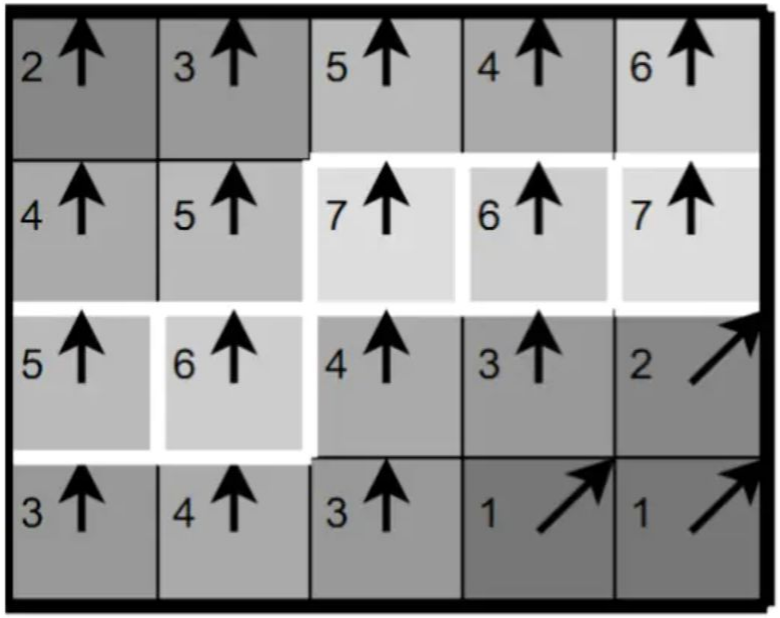
Non maximum Suppression

- Suppress the pixels in $|\nabla S|$ which are not local maximum



$$M(x,y) = \begin{cases} |\nabla S|(i,j) & if\,|\nabla S|(i,j) > |\nabla S|(i-1,j+1) \\ & and\ |\nabla S|(i,j) > |\nabla S|(i+1,j-1) \\ 0 & Otherwise \end{cases}$$

The Canny Edge Detector: Algorithm (step 3: Non-Max Suppression)……



Result after step 3

Gradient Magnitude

$M(x,y)$

Non-max Suppression

$g_N(x,y)$

The Canny Edge Detector: Algorithm
(step 4: Hysteresis Thresholding)

- Final operation is ***to threshold*** $\mathbf{g_N(x,y)}$ to reduce false edge points.
  - Non-max suppression outputs a more accurate representation of real edges in an image.
  - But you can see that some edges are more bright than others.
  - The brighter ones can be considered as strong edges but the lighter ones can actually be edges or they can be because of noise.
  - To solve the problem of "which edges are really edges and which are not" Canny uses the **Hysteresis Thresholding**.

The Canny Edge Detector: Algorithm (step 4: Hysteresis Thresholding)…

This steps attempts to improve on this situation by using two thresholds, a low threshold and a high threshold.

Hysteresis thresholding:

$$g_{NH}(x,y) = g_N(x,y) \geq T_H$$

$$g_{NL}(x,y) = g_N(x,y) \geq T_L$$

and

$$g_{NL}(x,y) = g_{NL}(x,y) - g_{NH}(x,y)$$

The ratio of the high and low threshold should be two or three to one.

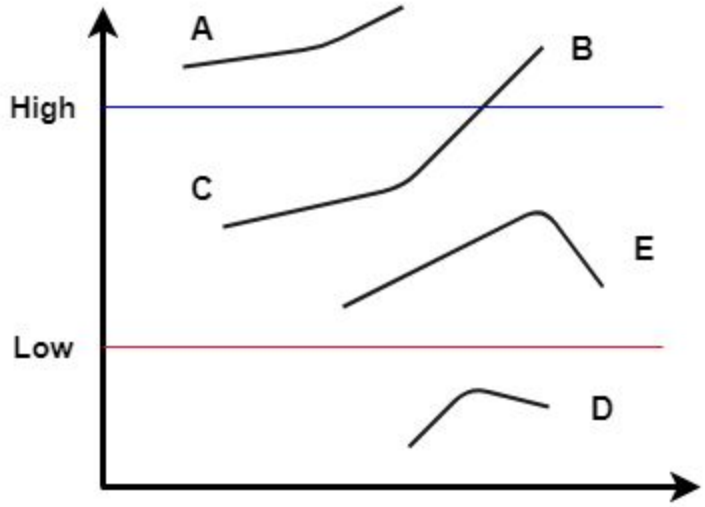
The Canny Edge Detector: Algorithm (step 4: Hysteresis Thresholding)…

- In Hysteresis thresholding, we set two thresholds ‘High’ and ‘Low’.
- If the threshold is set ***too low***, there will still be some false edge which is called ***false positive***.
- If the threshold is set ***too high***, then actual valid edge points will be eliminated which is called ***false negative***.

The Canny Edge Detector: Algorithm (step 5: connectivity analysis)

- Connectivity analysis to detect and link edges
- For the edge pixels values of $g_{NH}(x,y)$ and $g_{NL}(x,y)$.
  - Any edges with intensity greater than '**High**' are the **sure edges**.
  - Any edges with intensity less than '**Low**' are sure to be **non-edges**.
  - The edges **between 'High' and 'Low'** thresholds are classified as edges only **if they are connected to a sure edge** otherwise discarded.

Example



Here, A and B are sure-edges as they are above ‘High’ threshold. Similarly, D is a sure non-edge. Both ‘E’ and ‘C’ are weak edges but since ‘C’ is connected to ‘B’ which is a sure edge, ‘C’ is also considered as a strong edge. Using the same logic ‘E’ is discarded. This way we will get only the strong edges in the image..

This is based on the assumption that the edges are long lines.

The Canny Edge Detector: Algorithm (step 5) cont…

Depending on the value of T_H, the edges in $g_{NH}(x, y)$ typically have gaps. Longer edges are formed using the following procedure:

(a). Locate the next unvisited edge pixel, p, in $g_{NH}(x, y)$.

(b). Mark as valid edge pixel all the weak pixels in $g_{NL}(x, y)$ that are connected to p using 8-connectivity.

(c). If all nonzero pixel in $g_{NH}(x, y)$ have been visited go to step (d), esle return to (a).

(d). Set to zero all pixels in $g_{NL}(x, y)$ that were not marked as valid edge pixels.

200	**0**	**0**
0	0	**101**
0	0	0

gnH(x,y)

0	**0**	**57**
0	45	0
50	0	0

gnL(x,y)

200	**0**	**57**
0	45	**101**
0	0	0

g(x,y)

Final Result after step 5

Non-max Suppression

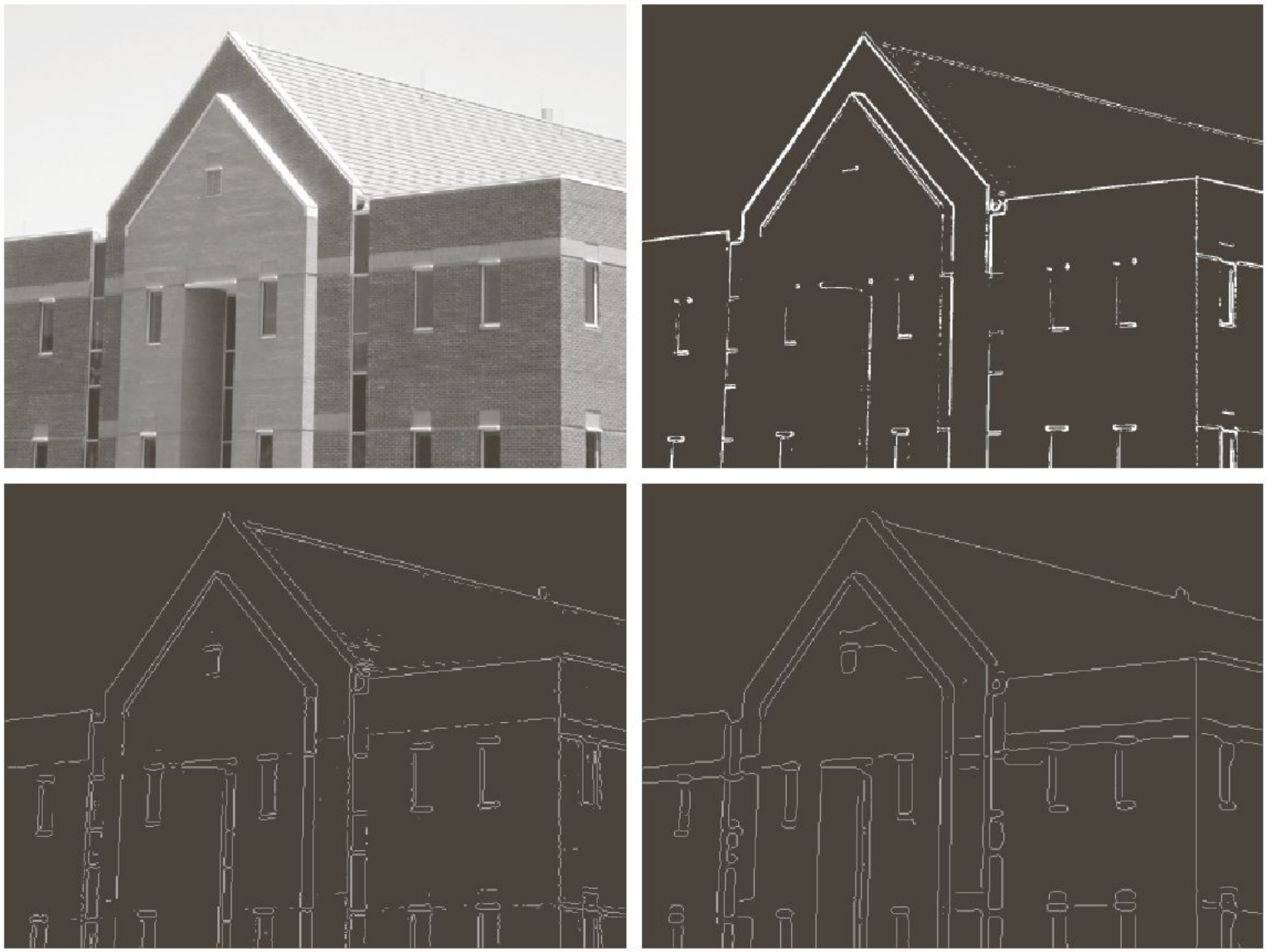
$g_N(x,y)$

Final Output

$g(x,y)$

The Canny Edge Detection: Summary

- Smooth the input image with a Gaussian filter
- Compute the gradient magnitude and angle images
- Apply nonmaxima suppression to the gradient magnitude image
- Use double thresholding and connectivity analysis to detect and link edges



a b
c d

FIGURE 10.25
(a) Original image of size 834 × 1114 pixels, with intensity values scaled to the range [0, 1].
(b) Thresholded gradient of smoothed image.
(c) Image obtained using the Marr-Hildreth algorithm.
(d) Image obtained using the Canny algorithm.
Note the significant improvement of the Canny image compared to the other two.

$T_L = 0.04; T_H = 0.10; \sigma = 4$ and a mask of size 25×25