Key Stages in Image Processing

1. Image Acquisition
2. Image Enhancement
3. Image Restoration
4. Color Image Processing
5. Wavelets and Other transforms
6. Compression
7. Morphological Processing
8. Segmentation
9. Feature Extraction
10. Image Pattern Classification

We may need only some of the steps in processing the digital image.

a. **Image Acquisition:**
   The image is captured by a sensor(eg: camera) and digitized if the output of the camera or sensor is not already in digital form. Using analog-to-digital converter.
b. **Image Enhancement:**
   The process of manipulating an image so that the result is more suitable than the original for specific applications, (brings out details and image features for the applications)
c. **Image Restoration:**
   The process of improving the appearance of an image. This mainly includes mathematical and probabilistic models of image degradation instead of human subjective preferences used in enhancement. (Photo taken in space)

d. **Color Image Processing:**
   Color image processing involves techniques and methods used to analyze, manipulate, and interpret images that contain color information. Unlike grayscale images, which only have varying shades

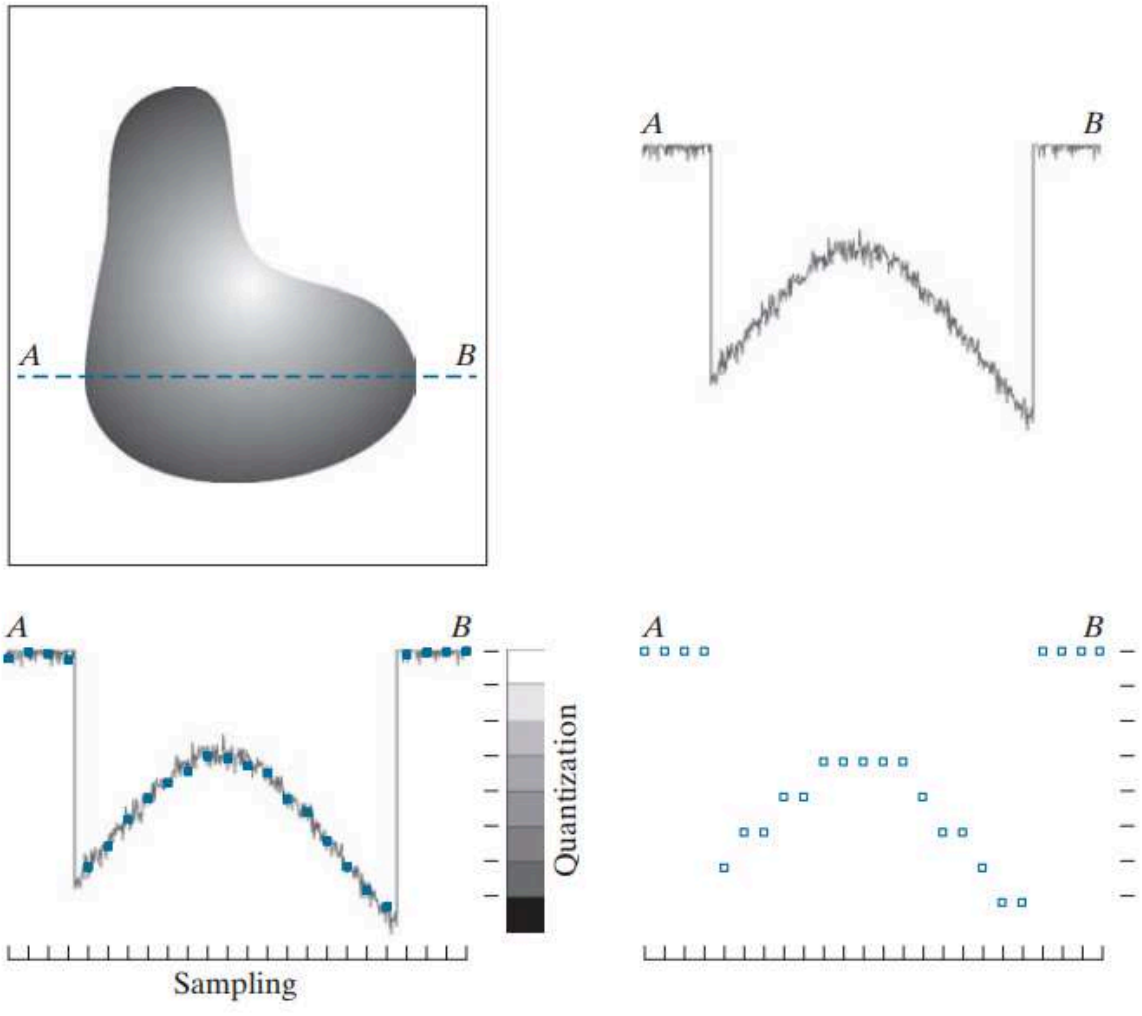
of gray, color images contain **multiple channels** representing different colors.

e. **Wavelets and Other transforms:**
   Wavelets and other transforms are fundamental tools in digital image processing used for analyzing, compressing, and enhancing images
f. **Compression:**
   It includes techniques for reducing the storage required for the same image or the bandwidth required to transmit it.
g. **Morphological Processing:**
   It deals with tools for extracting image components that are useful in the representations and descriptions of shapes. (fingerprint recognition)
h. **Segmentation (Hardest and has numerous algorithms):**
   Segmentation procedures partition an image into its constituent parts or objects. The more accurate the segmentation the more likely the recognition is to succeed.
i. **Feature extraction:**
   In image processing, it converts segmented image regions or boundaries into structured, quantitative information. It involves
   Feature Detection: Identifying key elements (e.g., edges, corners) in the image.
   Feature Description: Quantifying these elements with attributes like location, orientation, and size.
j. **Image pattern classification**:
   In digital image processing, **image pattern classification** involves assigning labels to image patterns based on their features. The approaches developed in this chapter are divided into three principal categories: classification by prototype matching, classification based on an optimal statistical formulation, and classification based on neural networks.

Sampling and Digitalization

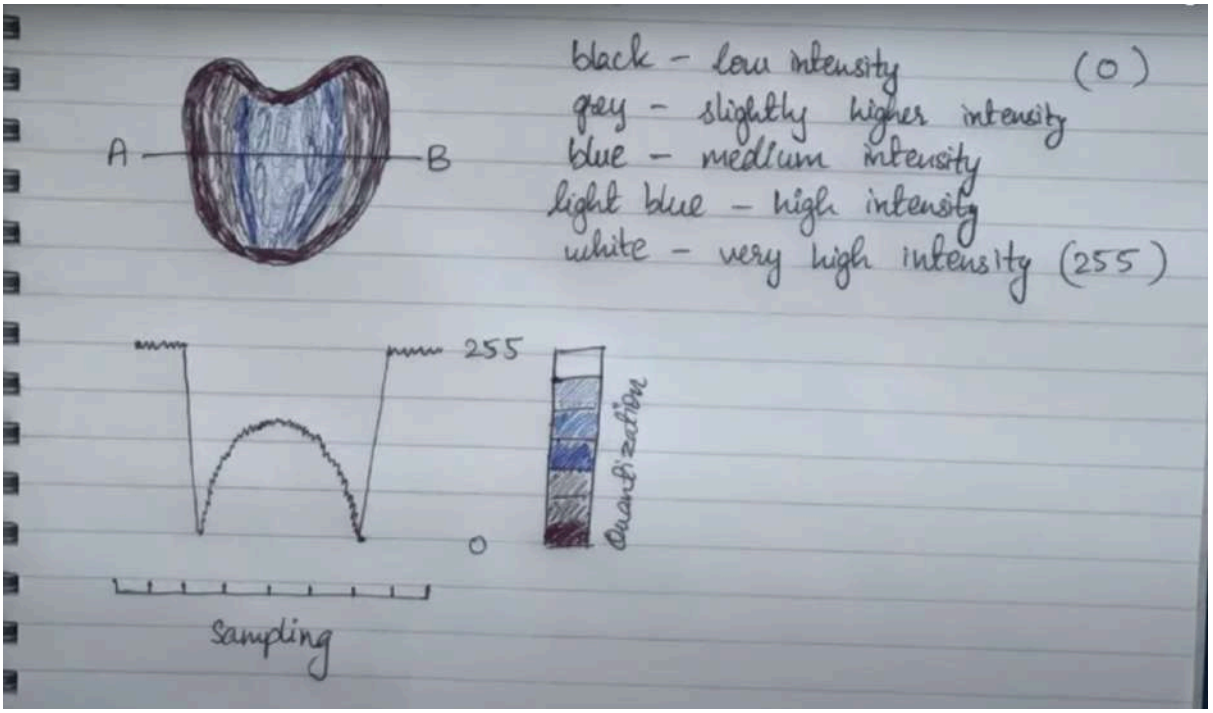
To become suitable for digital processing, an image function must be digitalized both spatially and in amplitude. This digitalization involves two processes

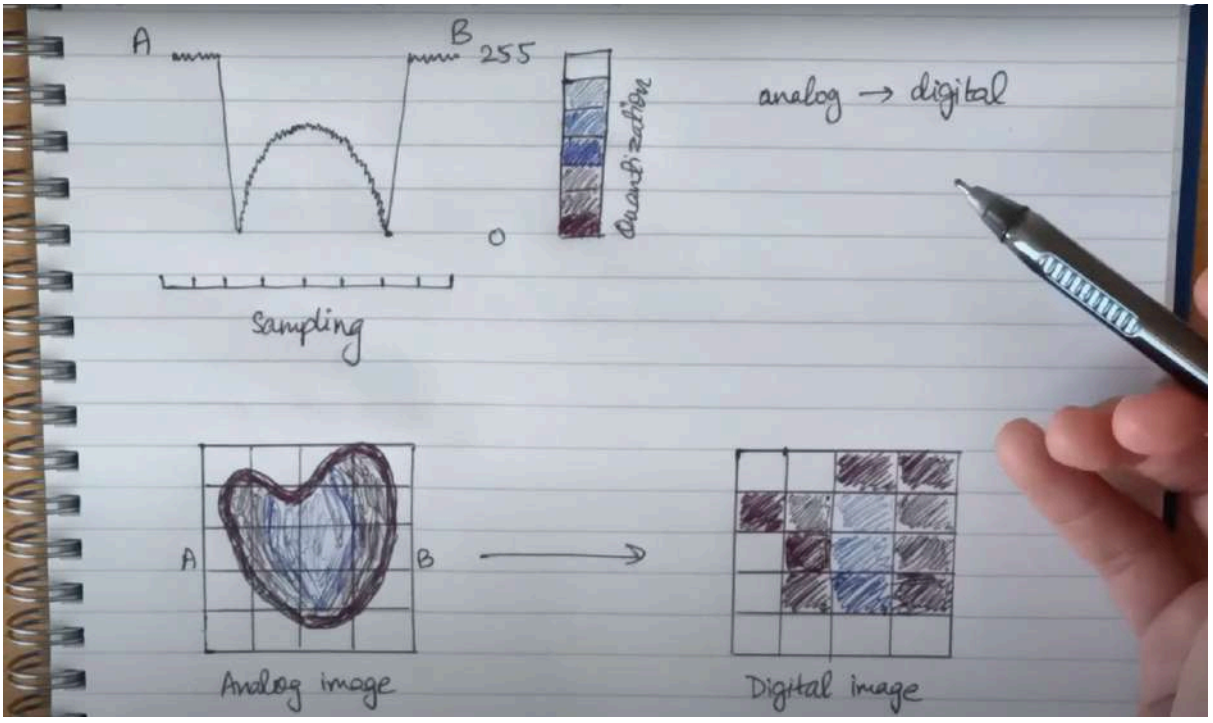
1. **Sampling** - **digitalizing** the **coordinate values** is called sampling
2. **Quantization** - **digitalizing** the **amplitude value** is called sampling



Black - low-intensity **0**
Gray - slightly higher intensity
Blue - medium intensity
Light Blue - high-intensity

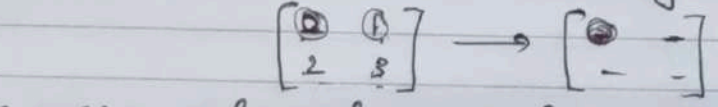
White -very high intensity **255**



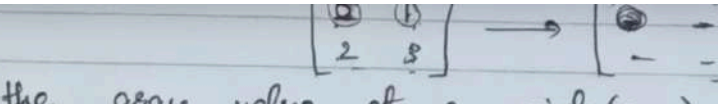


Point Operations

They are a method of image processing in which each pixel in the output image is only dependent upon the corresponding pixel in the input image and is independent of its location or neighbouring pixels.



Let 'r' be the gray value at a point (x, y) of the input image $f(x, y)$ and 's' be the gray value at a point (x, y) of the output image $g(x, y)$, then the point operation can be defined as :



Let 'r' be the gray value at a point (x, y) of the input image $f(x, y)$ and 's' be the gray value at a point (x, y) of the output image $g(x, y)$, then the point operation can be defined as :

$$s = T(r)$$

where T is the point operation of a certain gray-level mapping relationship between the original image and the output image.

1) Digital negative

$$S = (L-1) - r$$

4	3	5	2
3	6	4	6
2	2	6	5
7	6	4	1

Assuming the image to be 3-bit, we have $2^3 = 8$ levels.

$2^0 = 1$
$2^1 = 2$
$2^2 = 4$
$2^3 = 8$ 0-8

$n = 3$
$L = 2^n = 2^3 = 8$.

1) Digital negative

(0,7)

0 → 7

$$S = (L-1) - r$$
$$S = (8-1) - r$$
$$= 7 - r.$$

4	3	5	2
3	6	4	6
2	2	6	5
7	6	4	1

Assuming the image to be 3-bit, we have $2^3 = 8$ levels.

$r = 0, \quad S = 7 - 0 = 7.$
$r = 1, \quad S = 7 - 1 = 6$
$r = 2, \quad S = 7 - 2 = 5.$
$r = 7, \quad S = 7 - 7 = 0.$

$2^0 = 1$
$2^1 = 2$
$2^2 = 4$
$2^3 = 8$ 0-8

$n = 3$
$L = 2^n = 2^3 = 8$.

$s = (L-1) - r$

$s = (8-1) - r$

$= 7 - r.$

4	3	5	2
3	6	4	6
2	2	6	5
7	6	4	1

Assuming the image to be 3-bit, we have $2^3 = 8$ levels.

$2^0 = 1$

$2^1 = 2$

$2^2 = 4$

$2^3 = 8$ $\quad 0-8$

$r = 0, \; s = 7 - 0 = 7.$

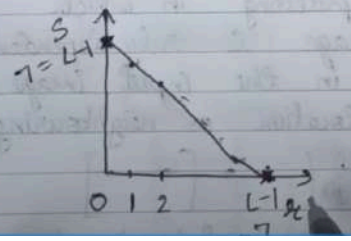
$r = 1, \; s = 7 - 1 = 6$

$r = 2, \; s = 7 - 2 = 5$

$[0, L-1]$ $\quad n = 3$

$r = 7, \; s = 7 - 7 = 0.$

$L = 2^n = 2^3 = 8$



3	4	2	5
4	1	3	1
5	5	1	2
0	1	3	6

Digital Negative

2) Thresholding with $T = 4$

$L = 8$

$L - 1 = 7$

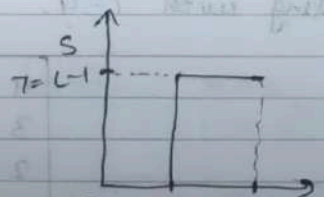
4	3	5	2
3	6	4	6
2	2	6	5
7	6	4	1

$$s = \begin{cases} L-1 = 7 & ; r \geq 4 \\ 0 & ; r < 4 \end{cases}$$

$$\begin{bmatrix} r = 0, 1, 2, 3 \rightarrow s = 0 \\ r = 4, 5, 6, 7 \rightarrow s = 7 \end{bmatrix}$$

7	0	7	0
0	7	7	7
0	0	7	7
7	7	7	0

Output image



3) Clipping with $r_1 = 2$ and $r_2 = 5$

$L = 8$

$L - 1 = 7$

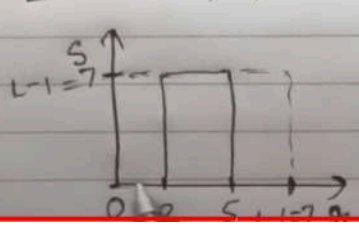
4	3	5	2
3	6	4	6
2	2	6	5
7	6	4	1

$$s = \begin{cases} L-1 = 7 & ; 2 \leq r \leq 5 \\ 0 & ; \text{otherwise} \end{cases}$$

$$\begin{bmatrix} r = 0, 1, 6, 7 \rightarrow S = 0 \\ r = 2, 3, 4, 5 \rightarrow s = 7 \end{bmatrix}$$

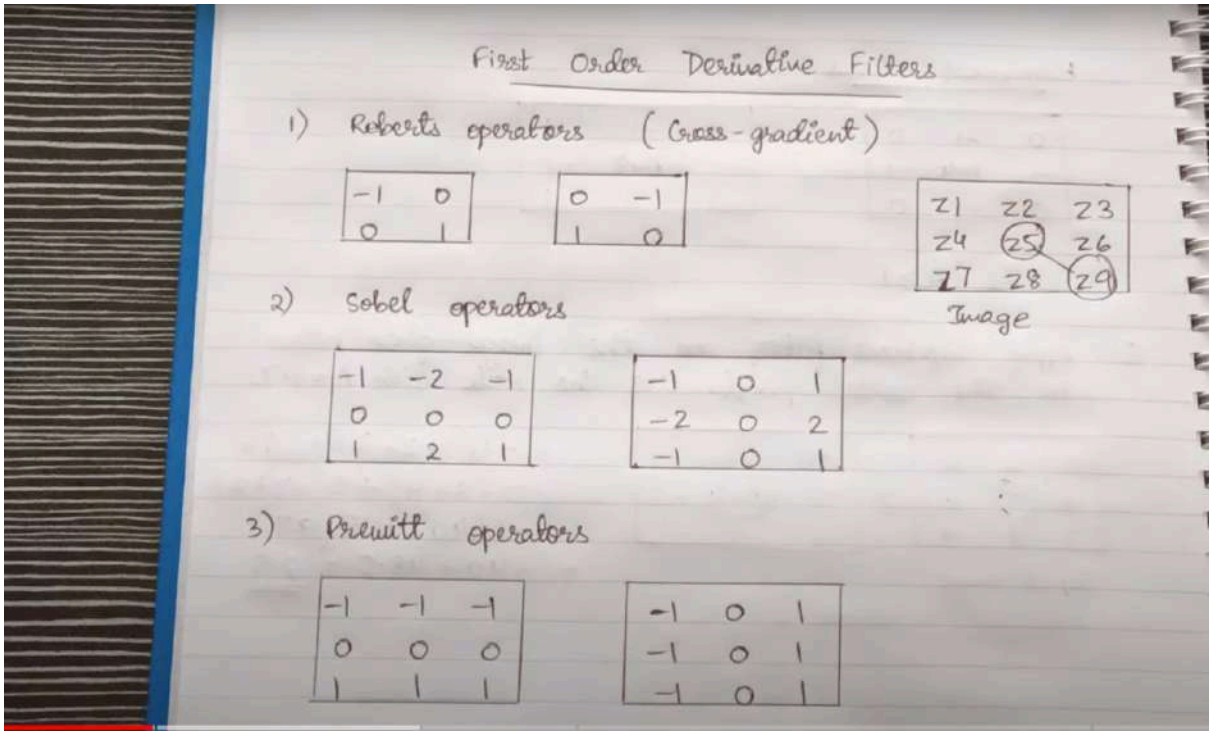
7	7	7	7
7	0	7	0
7	7	0	7
0	0	7	0

Output image



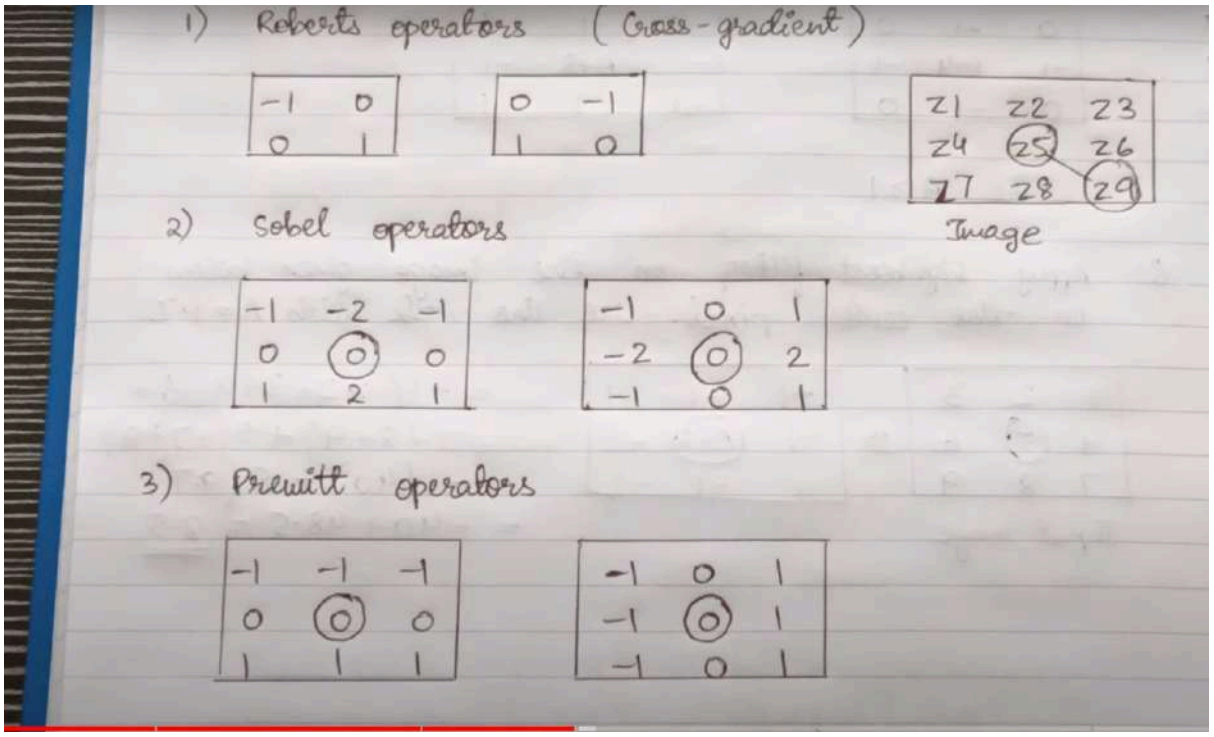
First Order Derivative Filters - Roberts, Sobel, and Prewitt

→ Operators are mainly used for edge detection in the below example we have 25 with the line 29 only diagonally so the 29 is the edge for the 25-pixel value.

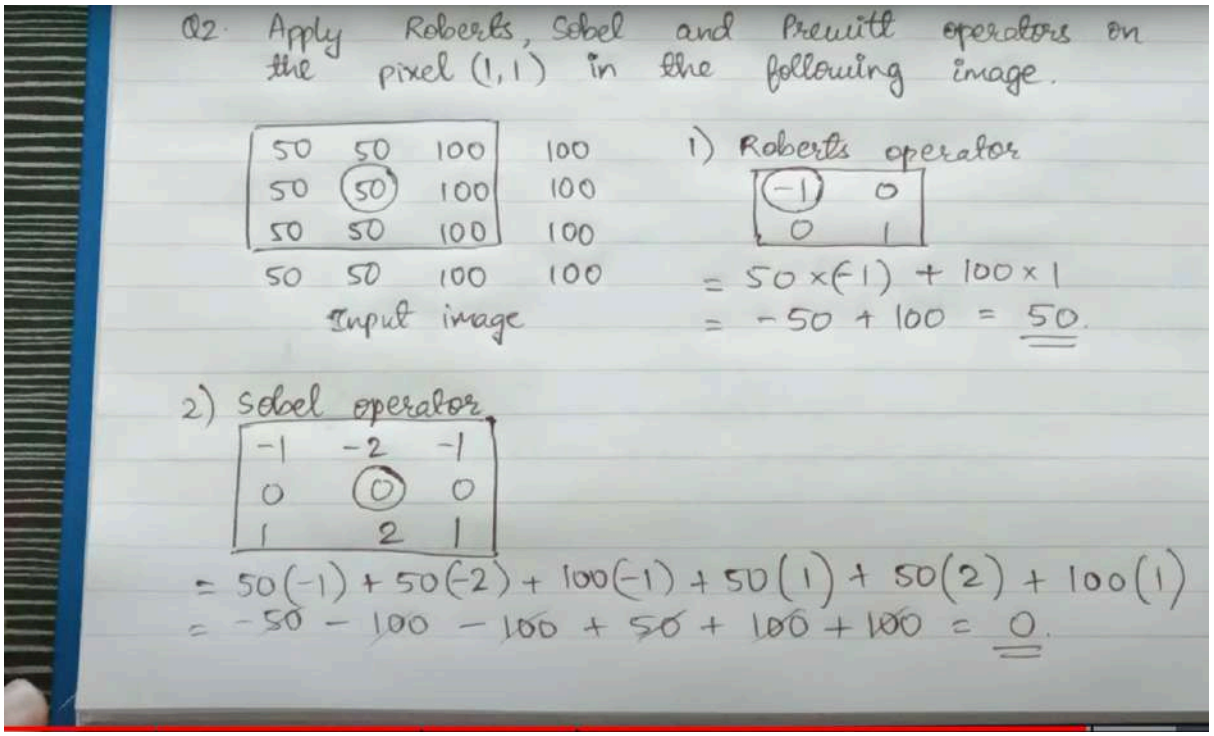


Sobel Operators and Prewitt Operators

- **Sobel operators** are involved in the **center pixel** only
- Whereas **Prewitt operators involve the center pixel** with **horizontal and vertical**



Example numerical:



3) Prewitt operator

50	50	100	100
50	(50)	100	100
50	50	100	100
50	50	100	100

Input image

$*$

-1	-1	-1
0	(0)	0
1	1	1

$= -1(50 + 50 + 100) + 1(50 + 50 + 100)$

$= \underline{0}.$

Grayscale Image

Consider the following 4x4 image matrix:

$$\text{Image} = \begin{bmatrix} 10 & 30 & 50 & 80 \\ 90 & 20 & 60 & 40 \\ 70 & 100 & 30 & 90 \\ 40 & 60 & 80 & 20 \end{bmatrix}$$

Prewitt Operator

The Prewitt operator uses the following kernels:

For detecting horizontal edges (G_x):

$$G_x = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}$$

For detecting vertical edges (G_y):

$$G_y = \begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}$$

We'll calculate the gradients at the pixel located at position (2, 2) (which is the value 30 in the matrix).

Applying the Prewitt Operator

Applying the Prewitt Operator

G_x Gradient:

We apply G_x to the 3x3 submatrix surrounding the pixel at (2, 2):

$$\begin{bmatrix} 20 & 60 & 40 \\ 100 & 30 & 90 \\ 60 & 80 & 20 \end{bmatrix}$$

Now, applying the G_x kernel:

$$G_x = (-1 \times 20) + (0 \times 60) + (1 \times 40) + (-1 \times 100) + (0 \times 30) + (1 \times 90) + (-1 \times 60$$

$$G_x = -20 + 0 + 40 - 100 + 0 + 90 - 60 + 0 + 20 = -30$$

G_y Gradient:

We apply G_y to the same submatrix:

$$G_y = (-1 \times 20) + (-1 \times 60) + (-1 \times 40) + (0 \times 100) + (0 \times 30) + (0 \times 90) + (1 \times 60$$

$$G_y = -20 - 60 - 40 + 0 + 0 + 0 + 60 + 80 + 20 = 40$$

Magnitude of the Gradient

The magnitude of the gradient is calculated as:

$$\text{Magnitude} = \sqrt{G_x^2 + G_y^2} = \sqrt{(-30)^2 + (40)^2} = \sqrt{900 + 1600} = \sqrt{2500} = 50$$

Summary

For the given 4x4 image, the Prewitt operator detects an edge with a gradient magnitude of **50** at the pixel located at (2, 2).

First-order derivative filters are used in image processing to detect edges by emphasizing areas where the intensity of the image changes sharply. The Roberts, Sobel, and Prewitt operators are among the most commonly used first-order derivative filters.

1. Roberts Operator

- **Purpose:** Detects edges by calculating the gradient of the image intensity at each point.
- **Method:** Uses a pair of 2x2 convolution kernels:

$$G_x = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}, \quad G_y = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}$$

- **Output:** The magnitude of the gradient is computed as:

$$\text{Magnitude} = \sqrt{G_x^2 + G_y^2}$$

- **Use Case:** Suitable for detecting diagonal edges, but sensitive to noise.

2. Sobel Operator

- **Purpose:** Identifies edges by approximating the gradient of the image using larger convolution kernels, which also smooth the image to reduce noise.
- **Method:** Uses 3x3 convolution kernels:

$$G_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}, \quad G_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$

- **Output:** Similar to the Roberts operator, the magnitude of the gradient is:

$$\text{Magnitude} = \sqrt{G_x^2 + G_y^2}$$

- **Use Case:** More accurate edge detection, especially for horizontal and vertical edges.

3. Prewitt Operator

- **Purpose:** Similar to the Sobel operator but with slightly different convolution kernels.

3. Prewitt Operator

- **Purpose:** Similar to the Sobel operator but with slightly different convolution kernels.
- **Method:** Also uses 3x3 convolution kernels:

$$G_x = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}, \quad G_y = \begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}$$

- **Output:** The magnitude of the gradient is calculated as:

$$\text{Magnitude} = \sqrt{G_x^2 + G_y^2}$$

- **Use Case:** Used for detecting edges, particularly when computational efficiency is required over accuracy.

Summary:

- **Roberts:** Diagonal edges, sensitive to noise.
- **Sobel:** Horizontal/vertical edges, noise-resistant.
- **Prewitt:** Simpler than Sobel, less sensitive to noise, slightly less accurate.

These operators are foundational for edge detection in image processing, with each having specific strengths depending on the application.

provide numerical example

Let's walk through a numerical example using a small 3x3 grayscale image and apply the Roberts, Sobel, and Prewitt operators to detect edges.

Grayscale Image

Consider the following 3x3 image matrix:

Grayscale Image

Consider the following 3x3 image matrix:

$$\text{Image} = \begin{bmatrix} 200 & 200 & 200 \\ 200 & 100 & 200 \\ 200 & 200 & 200 \end{bmatrix}$$

1. Roberts Operator

The Roberts operator uses the following kernels:

$$G_x = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}, \quad G_y = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}$$

Applying G_x and G_y:

At the central pixel (100), the Roberts operator computes gradients as:

$$G_x = (200 \times 1) + (100 \times 0) + (200 \times 0) + (200 \times -1) = 200 - 200 = 0$$

$$G_y = (200 \times 0) + (200 \times 1) + (200 \times -1) + (100 \times 0) = 200 - 200 = 0$$

Magnitude of Gradient:

$$\text{Magnitude} = \sqrt{G_x^2 + G_y^2} = \sqrt{0^2 + 0^2} = 0$$

The Roberts operator detects no edge at this location.

2. Sobel Operator

The Sobel operator uses the following kernels:

$$G_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}, \quad G_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$

Applying G_x and G_y:

At the central pixel (100), the Sobel operato ↓ mputes gradients as:

The Sobel operator uses the following kernels:

$$G_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}, \quad G_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$

Applying G_x and G_y:

At the central pixel (100), the Sobel operator computes gradients as:

$$G_x = (-1 \times 200) + (0 \times 200) + (1 \times 200) + (-2 \times 200) + (0 \times 100) + (2 \times 200) + (-$$

$$G_x = (-200 + 0 + 200 - 400 + 0 + 400 - 200 + 0 + 200) = 0$$

$$G_y = (-1 \times 200) + (-2 \times 200) + (-1 \times 200) + (0 \times 200) + (0 \times 100) + (0 \times 200) + (1$$

$$G_y = (-200 - 400 - 200 + 0 + 0 + 0 + 200 + 400 + 200) = 0$$

Magnitude of Gradient:

$$\text{Magnitude} = \sqrt{G_x^2 + G_y^2} = \sqrt{0^2 + 0^2} = 0$$

The Sobel operator also detects no edge at this location.

3. Prewitt Operator

The Prewitt operator uses the following kernels:

$$G_x = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}, \quad G_y = \begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}$$

Applying G_x and G_y:

At the central pixel (100), the Prewitt operator computes gradients as:

$$G_x = (-1 \times 200) + (0 \times 200) + (1 \times 200) + (-1 \times 200) + (0 \times 100) + (1 \times 200) + (-$$

$$G_x = (-200 + 0 + 200 - 200 + 0 + 200 - 200 + 0 + 200) = 0$$

$$G_y = (-1 \times 200) + (-1 \times 200) + (-1 \times 200) + (0 \times 200) + (0 \times 100) + (0 \times 200) + (1$$

Grayscale Image

Consider the following 4x4 image matrix:

$$\text{Image} = \begin{bmatrix} 12 & 56 & 34 & 78 \\ 90 & 15 & 60 & 22 \\ 45 & 88 & 25 & 67 \\ 98 & 54 & 33 & 11 \end{bmatrix}$$

Roberts Operator

The Roberts operator uses the following kernels:

$$G_x = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}, \quad G_y = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}$$

Let's calculate the gradients at the pixel located at position (2, 2) (which is the value 25 in the matrix).

Applying the Roberts Operator

G_x Gradient:

We apply G_x to the submatrix surrounding the pixel at (2, 2):

$$\begin{bmatrix} 15 & 60 \\ 88 & 25 \end{bmatrix}$$

$$G_x = (15 \times 1) + (60 \times 0) + (88 \times 0) + (25 \times -1) = 15 - 25 = -10$$

G_y Gradient:

We apply G_y to the same submatrix:

$$G_y = (15 \times 0) + (60 \times 1) + (88 \times -1) + (25 \times 0) = 60 - 88 = -28$$

Magnitude of the Gradient

The magnitude of the gradient is calculated as:

$$\text{Magnitude} = \sqrt{G_x^2 + G_y^2} = \sqrt{(-10)^2 + (-28)^2} = \sqrt{100 + 784} = \sqrt{884} \approx 29.7$$

Summary

For the given 4x4 image, the Roberts operator detects an edge with a gradient magnitude of approximately **29.7** at the pixel located at (2, 2).

Region-Based Segmentation

- Sometimes edges and thresholds do not provide good results in edge detection
- Region-based segmentation is based on the connectivity of similar pixels in a region

There are two main approaches for the region-based segmentation:

1. Region-Splitting
2. Region-Growing

Algorithms or the steps/procedure in Region-Growing

1. Find the connected components in S(x,y) and reduce each connected component to one pixel. Label all such pixels found as 1. All other pixels in S are labeled 0.

2. Form an image Fq such that at each point (x,y) , fq(x,y) = 1 if the input image satisfies a given predicate Q at those co-ordinates and Fq(x,y) = 0 otherwise,

3. Let g be an image formed by appending to each seed point in S all the I-valued points in fq that are 8-Connected to that seed point.

4. Label each connected component in g with a different region label (Ex. integers or letters). This is the segmented image obtained by region-growing.

Algorithms or the steps/procedure in Region-Splitting and Merging Techniques

1. If a region R is inhomogeneous (P(R) = False), then R is split into four sub-regions
2. If 2 adjacent regions Ri, Rj are homogeneous (P(Ri U Rj) = True), then they are merged.
3. The algorithm stops when no further splitting or merging is possible

Note: Condition for region growing:
abs(seed value - pixel value) <= Threshold

Q1. Apply region growing on the following image with initial point at (2,2) and threshold value as 2. Use 4 connectivity.

T = 2

	0	1	2	3
0	0	1	2	0
1	2	5	6	1
2	1	4	(7)	3
3	0	2	5	1

Ans. The segmented region is shown in the following figure.
Condition → absolute difference <= 2.
4 way connectivity

Q1. Apply region growing on the following image with initial point at (2,2) and threshold value as 2. Use 4 connectivity.

T = 2

	0	1	2	3
0	0	1	2	0
1	2	5[a]	6[a]	1
2	1	4	7[a]	3
3	0	2	5[a]	1

a

7 - 6 = 1 ✓
7 - 4 = 3 ✗
7 - 5 = 2 ✓
7 - 3 = 4 ✗

Ans. The segmented region is shown in the following figure.
Condition → absolute difference <= 2.
4 way connectivity

Ans: The segmented region is shown in the following figure.

Condition → absolute difference <= 2.
4 way connectivity

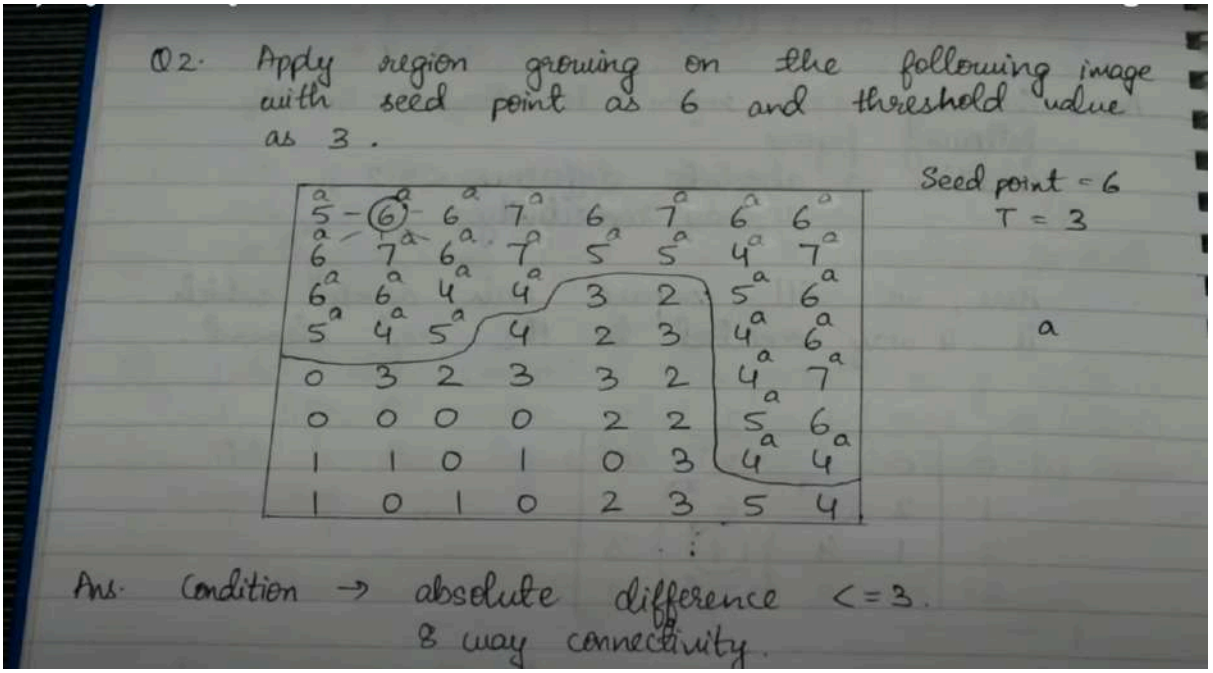
Here, we will compare each element which is 4 way connected to the seed element.

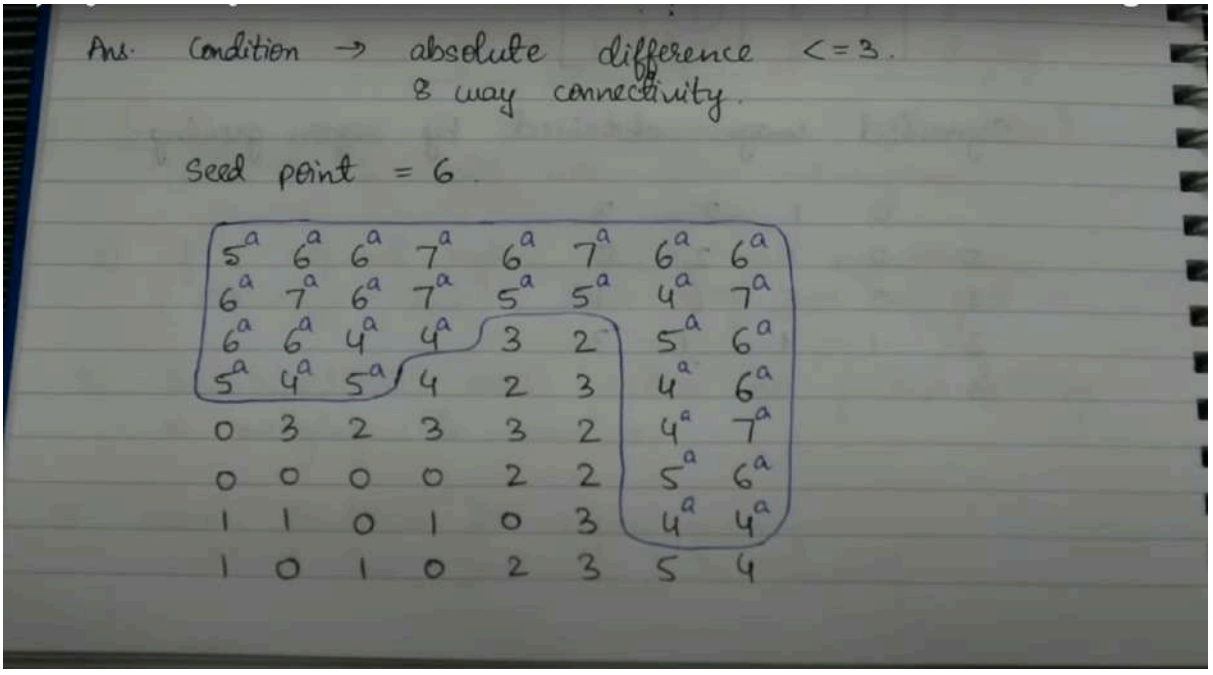
	0	1	2	3
0	0	1	2	0
1	2	5[a]	6[a]	1
2	1	4	7[a]	3
3	0	2	5[a]	1

Segmented image obtained from region growing:

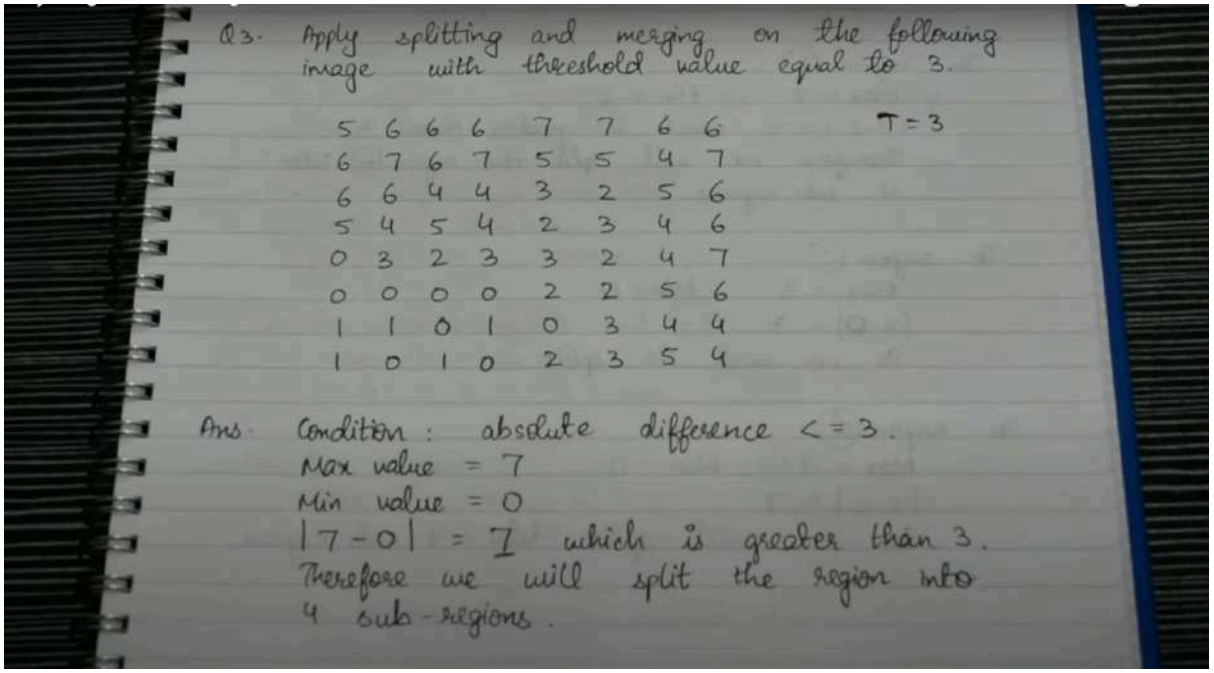
	0	1	2	3
0	0	0	0	0
1	0	1	1	0
2	0	0	1	0
3	0	0	1	0

Question:

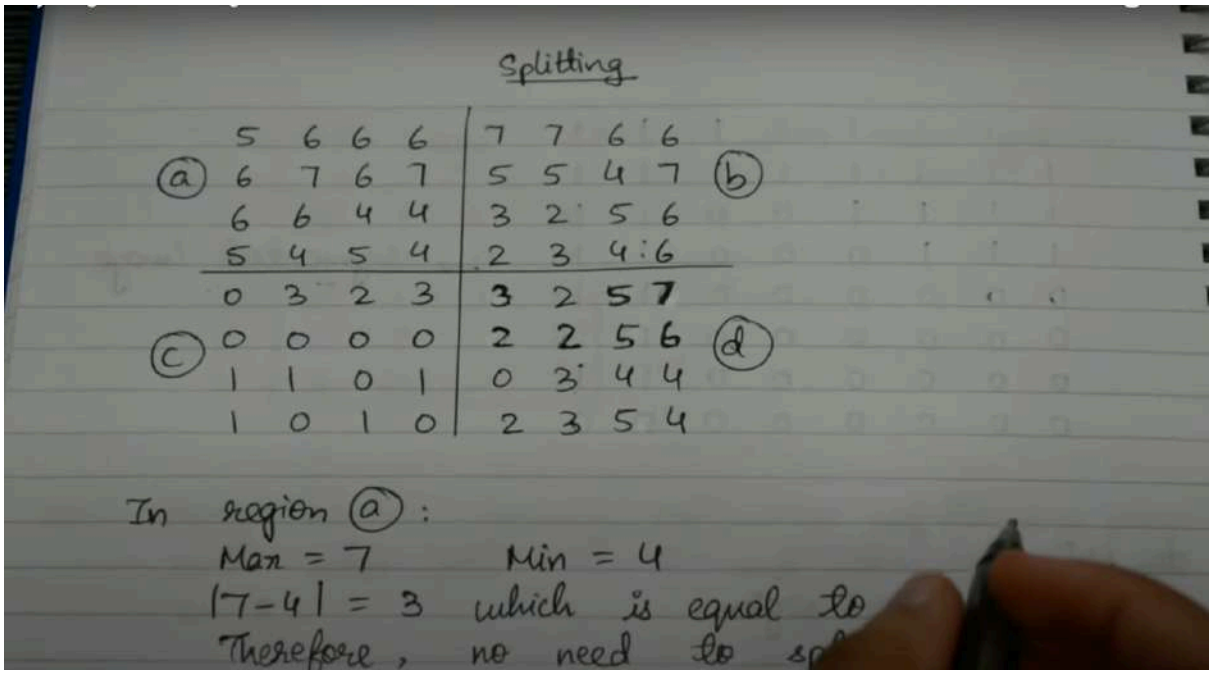

Q2. Apply region growing on the following image with seed point as 6 and threshold value as 3.
Seed point = 6
T = 3
5 6 6 7 6 7 6 6
6 7 6 7 5 5 4 7
6 6 4 4 3 2 5 6
5 4 5 4 2 3 4 6
0 3 2 3 3 2 4 7
0 0 0 0 2 2 5 6
1 1 0 1 0 3 4 4
1 0 1 0 2 3 5 4
a
Ans. Condition → absolute difference <= 3.
8 way connectivity.


Ans. Condition → absolute difference <= 3.
8 way connectivity.
Seed point = 6.
5 6 6 7 6 7 6 6
6 7 6 7 5 5 4 7
6 6 4 4 3 2 5 6
5 4 5 4 2 3 4 6
0 3 2 3 3 2 4 7
0 0 0 0 2 2 5 6
1 1 0 1 0 3 4 4
1 0 1 0 2 3 5 4

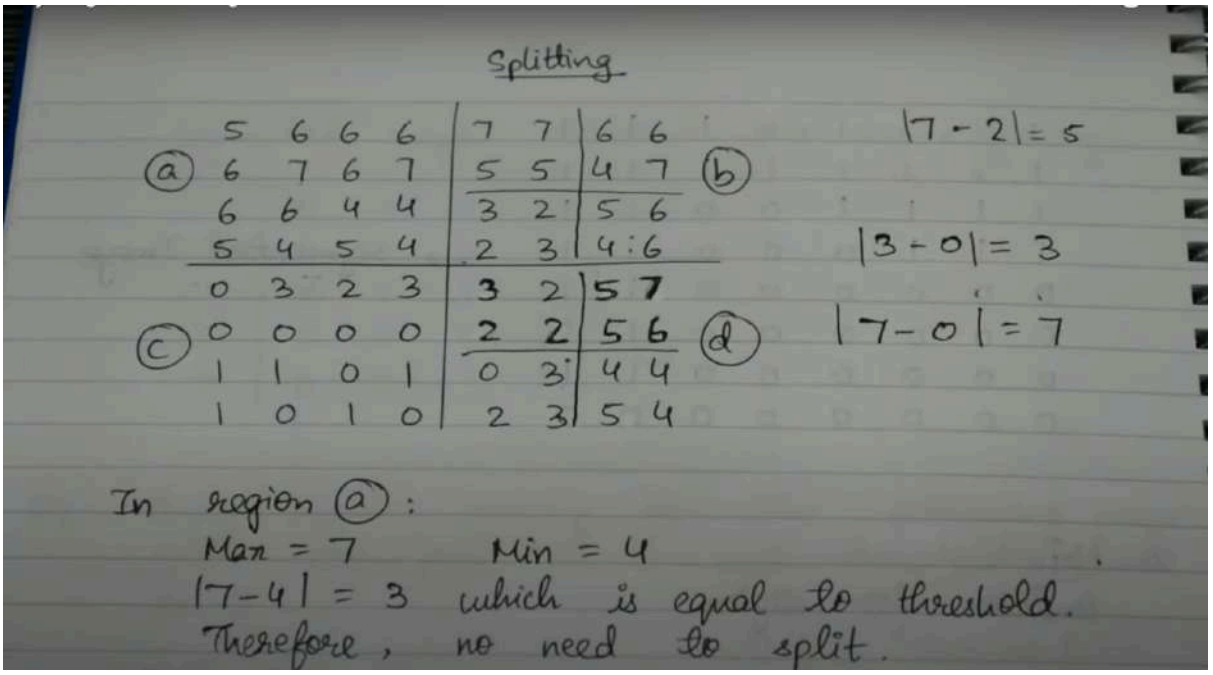
Region Splitting and Merging



Splitting



Sub-Regions Sub-Parts



Splitting

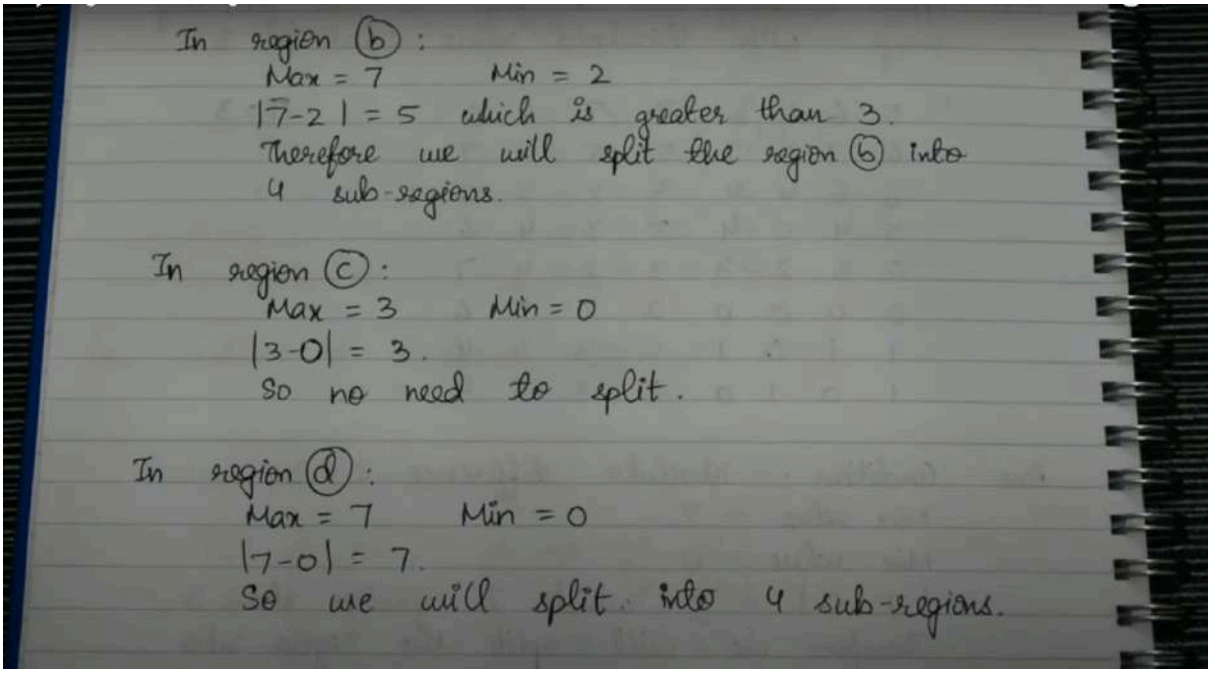
	5	6	6	6	7	7	6	6
(a)	6	7	6	7	5	5	4	7 (b)
	6	6	4	4	3	2	5	6
	5	4	5	4	2	3	4	6
	0	3	2	3	3	2	5	7
(c)	0	0	0	0	2	2	5	6 (d)
	1	1	0	1	0	3	4	4
	1	0	1	0	2	3	5	4

$|7-2| = 5$

$|3-0| = 3$

$|7-0| = 7$

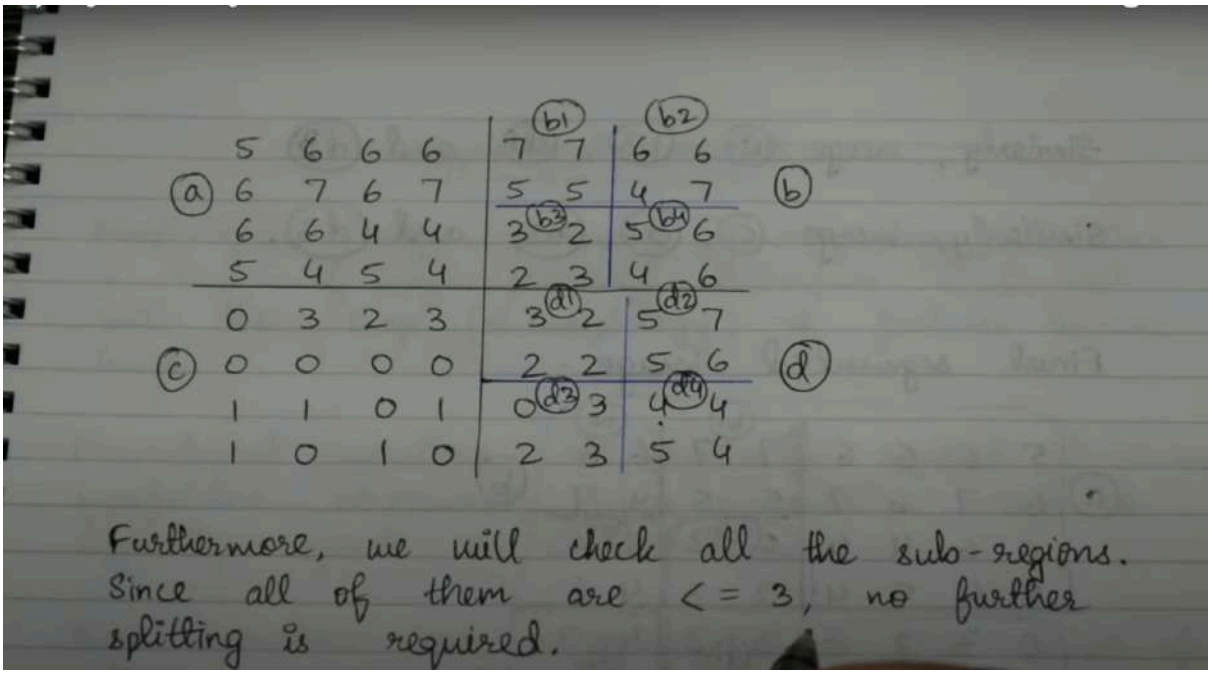
In region (a):
Max = 7 Min = 4
$|7-4| = 3$ which is equal to threshold.
Therefore, no need to split.



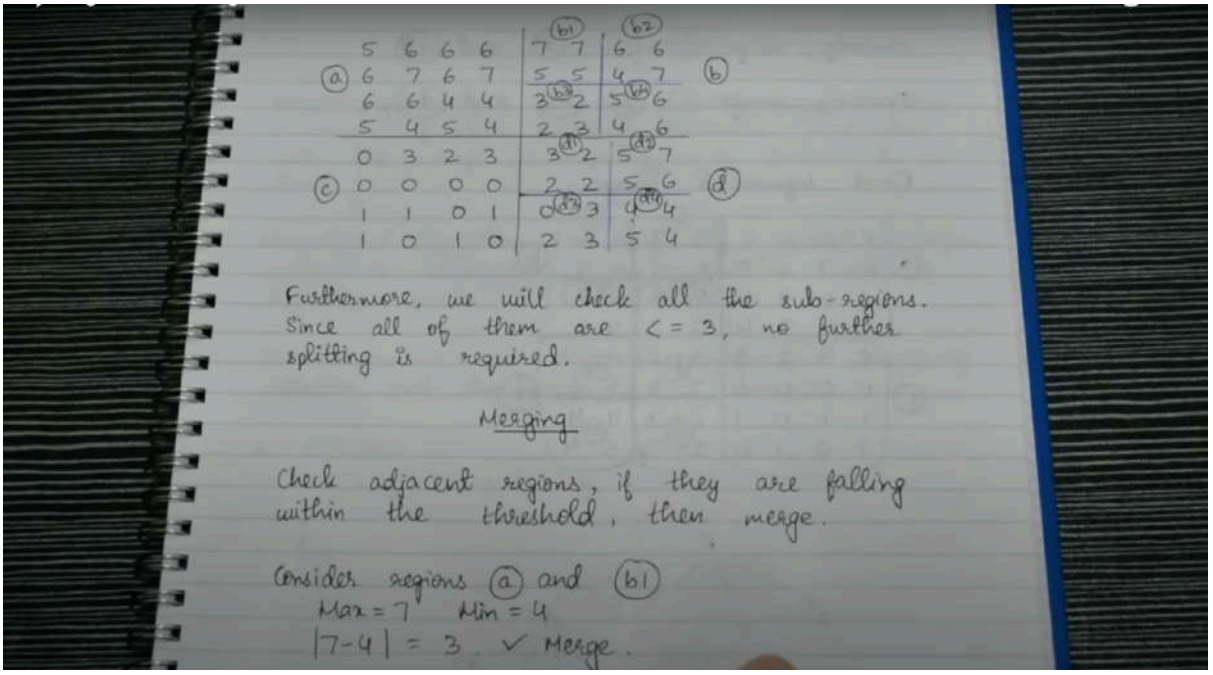
In region (b):
Max = 7 Min = 2
$|7-2| = 5$ which is greater than 3.
Therefore we will split the region (b) into 4 sub-regions.

In region (c):
Max = 3 Min = 0
$|3-0| = 3$.
So no need to split.

In region (d):
Max = 7 Min = 0
$|7-0| = 7$.
So we will split into 4 sub-regions.



Furthermore, we will check all the sub-regions. Since all of them are $<= 3$, no further splitting is required.



Furthermore, we will check all the sub-regions. Since all of them are $<= 3$, no further splitting is required.

Merging

Check adjacent regions, if they are falling within the threshold, then merge.

Consider regions (a) and (b1)

Max = 7 Min = 4

$|7-4| = 3$. ✓ Merge.

Consider regions (a b1) and (b2)
Max = 7 Min = 4
|7-4| = 3. ✓ Merge.

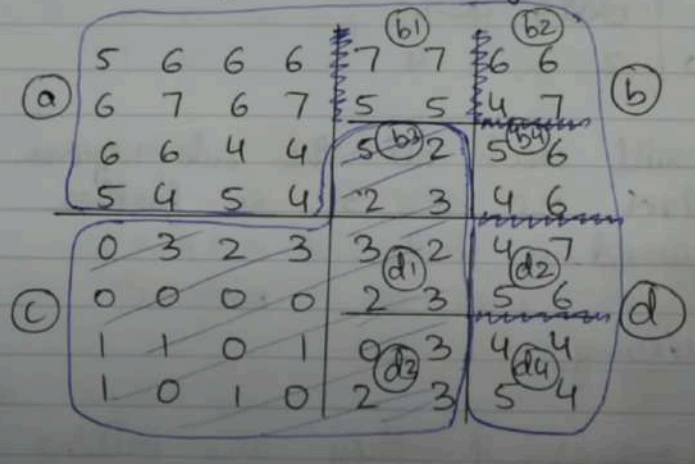
Consider regions (a b1 b2) and (b4)
Max = 7 Min = 4
|7-4| = 3 ✓ Merge

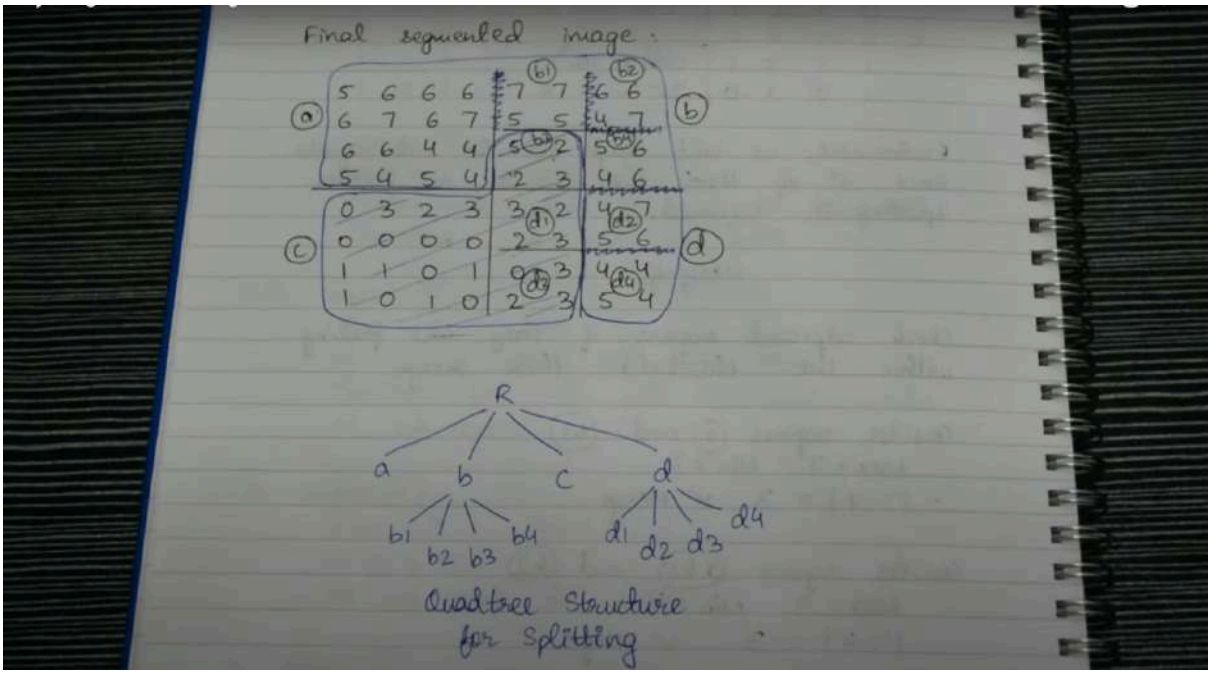
Consider regions (a b1 b2 b4) and d2
Max = 7 Min = 4
|7-4| = 3 ✓ Merge.

Similarly, merge (a b1 b2 b4 d2) with (d4).

Similarly, merge (c), (d1), (b3) and (d3).

Final segmented image:





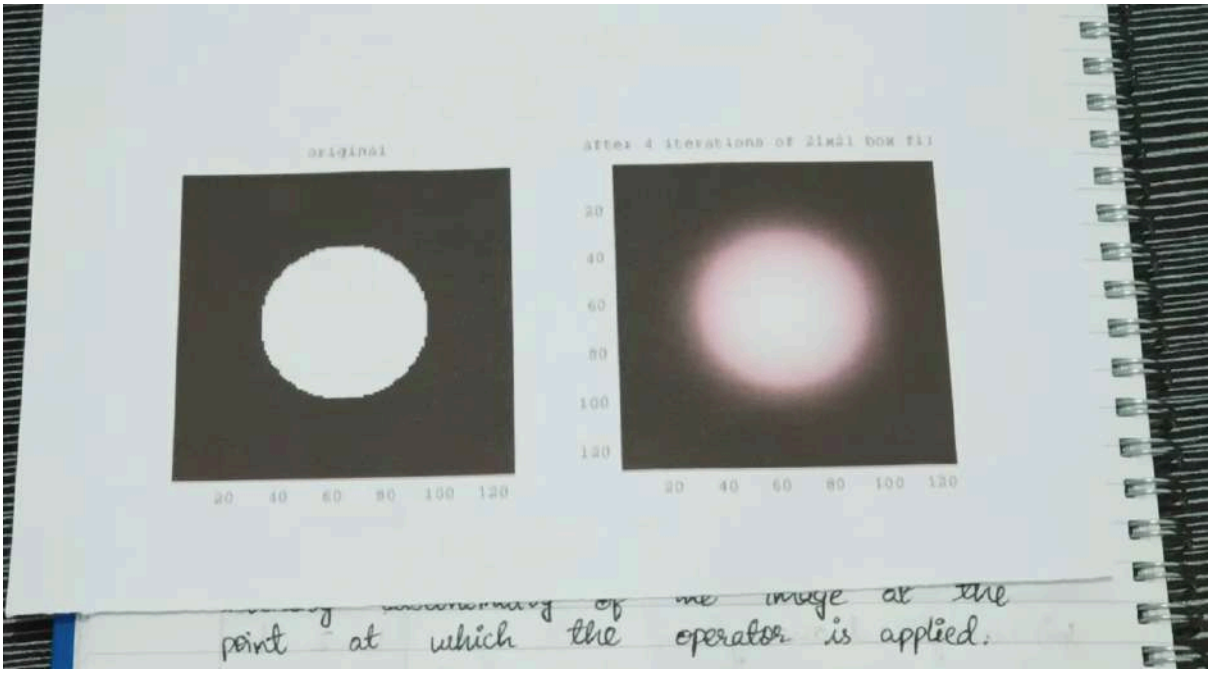
Sharpening Spatial filters in digital image processing with examples

→ The **principle objective** of the **sharpening** is to **highlight transitions** in **intensity**.

→ Applications of image sharpening include electronic printing, medical imaging, industrial inspection, and autonomous guidance in military systems.

Blurring → pixel averaging
Sharpening → spatial differentiation

→ The strength of the response of a **derivation operator** is **proportional** to the degree of intensity discontinuity of the image at the point at which the operator is applied.



point at which the operator is applied.

Foundation of sharpening filters

1) First-order derivative of a one-dimensional function $f(x)$:

$$\frac{\partial f}{\partial x} = f(x+1) - f(x)$$

2) Second-order derivative of a one-dimensional function $f(x)$:

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

Laplacian filter

→ It highlights gray-level discontinuities in an image.

→ It deemphasizes regions with slowly varying gray levels.

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

$f(x)$

$f(x,y)$

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

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

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

* Laplacian mask

0	1	0
1	-4	1
0	1	0

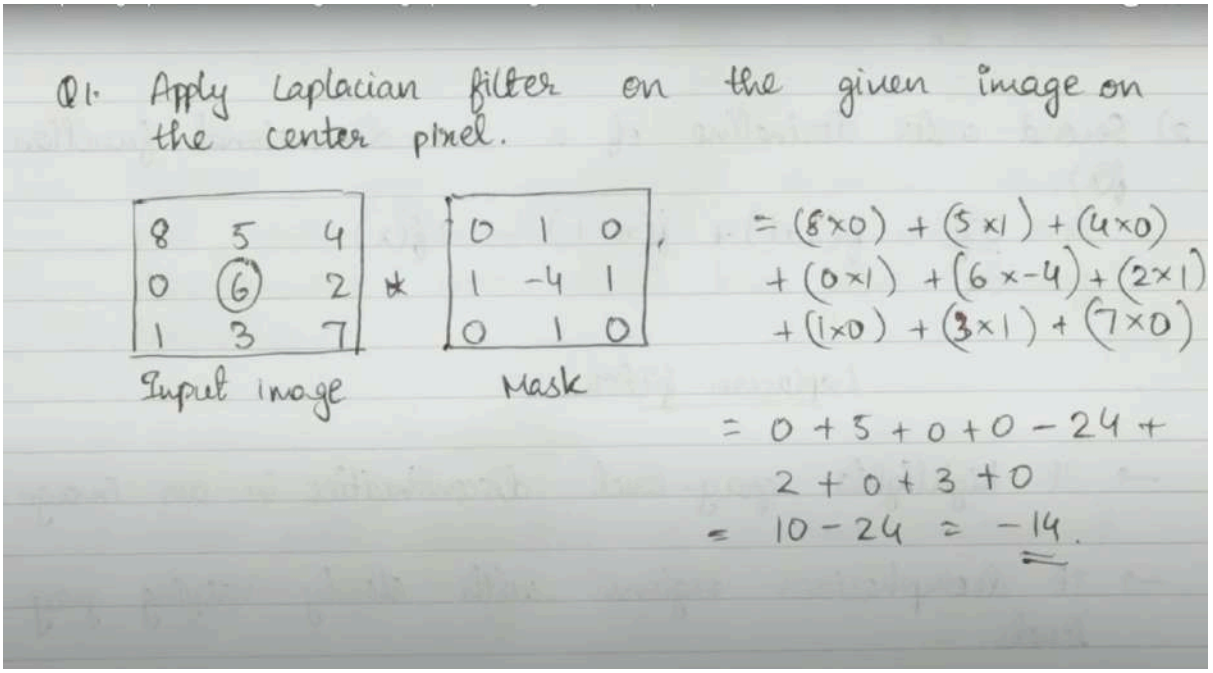
1	1	1
1	-8	1
1	1	1

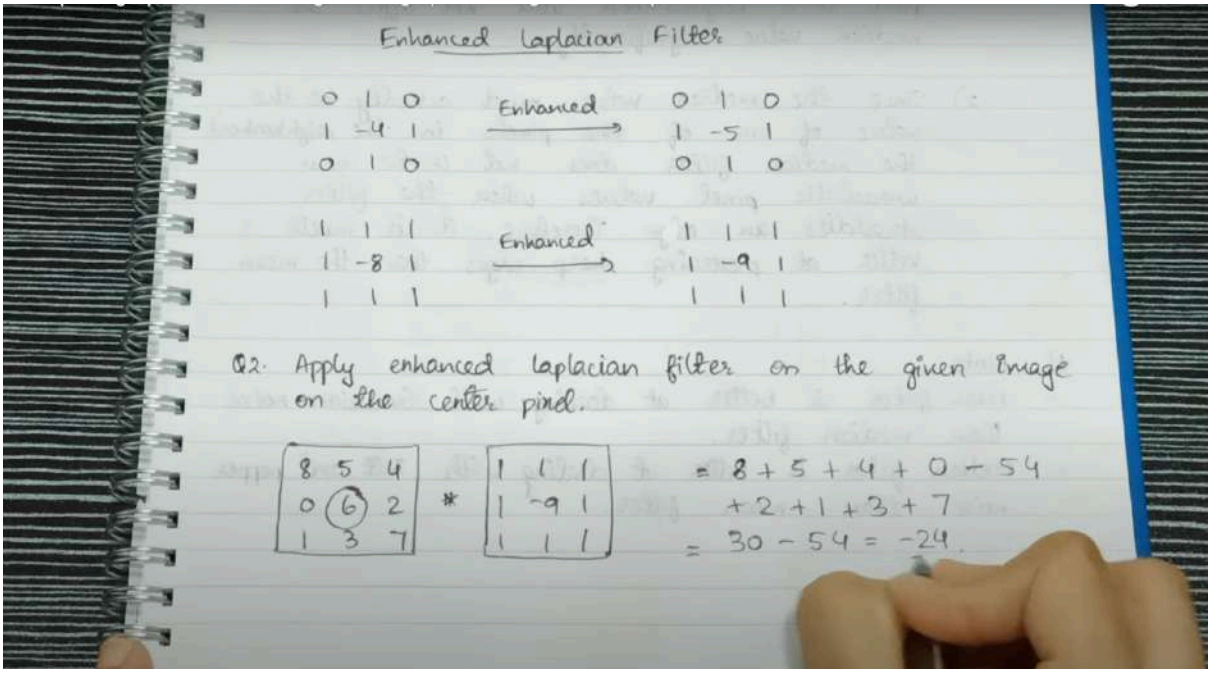
0	-1	0
-1	4	-1
0	-1	0

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

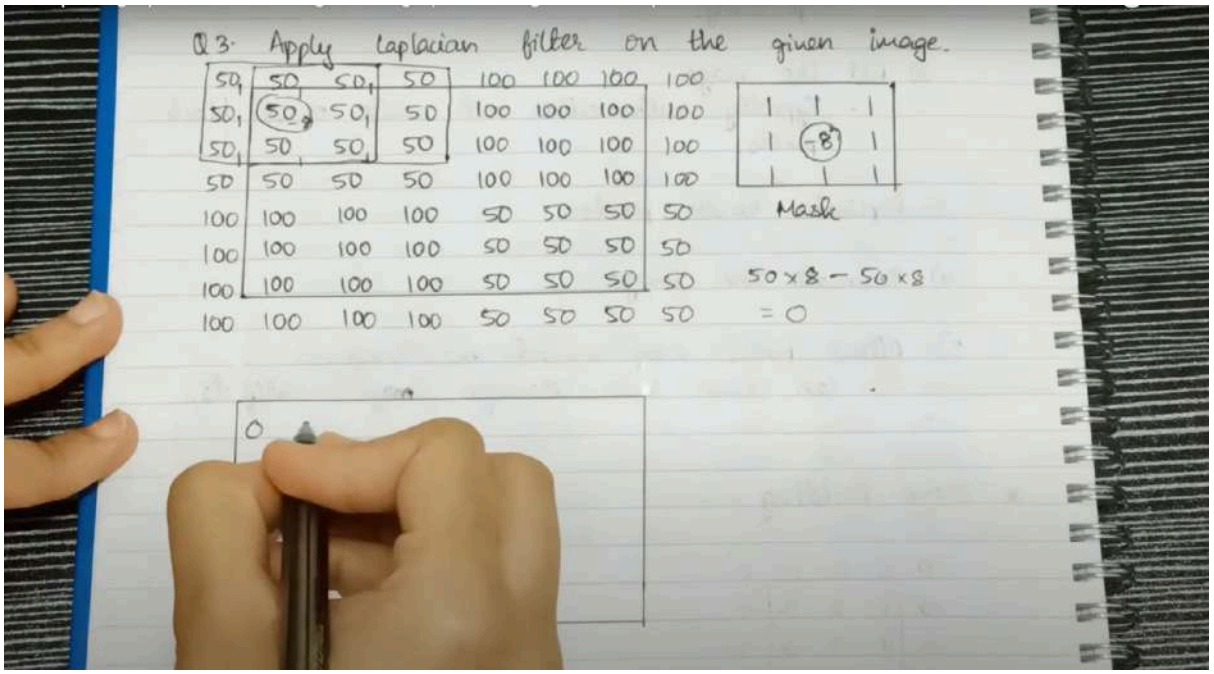
$f(x-1, y-1)$	$f(x, y-1)$	$f(x+1, y-1)$
$f(x-1, y)$	$f(x, y)$	$f(x+1, y)$
$f(x-1, y+1)$	$f(x, y+1)$	$f(x+1, y+1)$

Input image

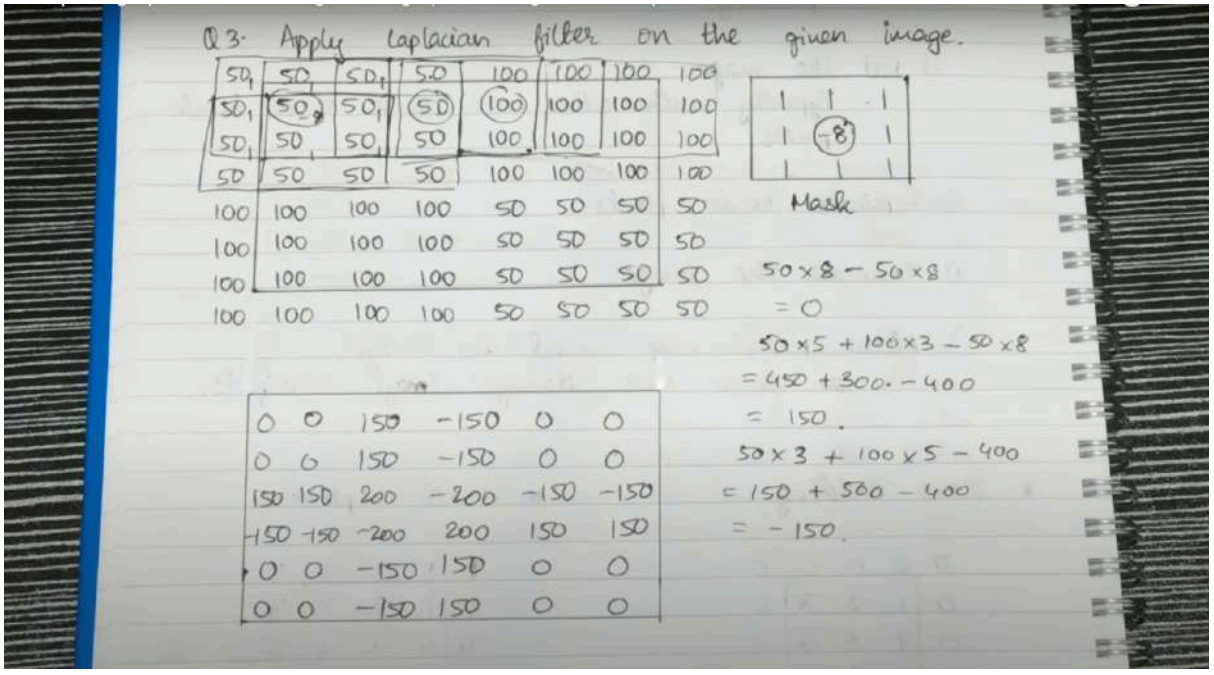




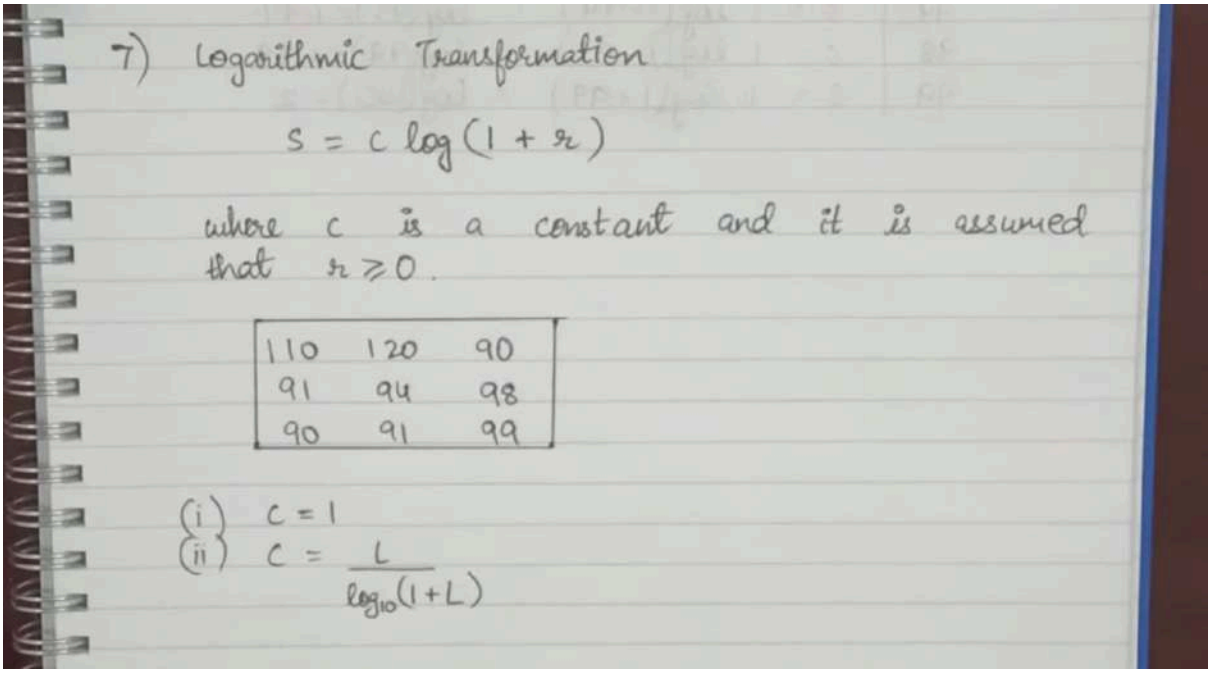
Note: we have a total of eight 50 and one is -8 so for each ease we do
= 50 x 8 + 50 x -8
= 0
Since it is a 3 x 3 matrix so

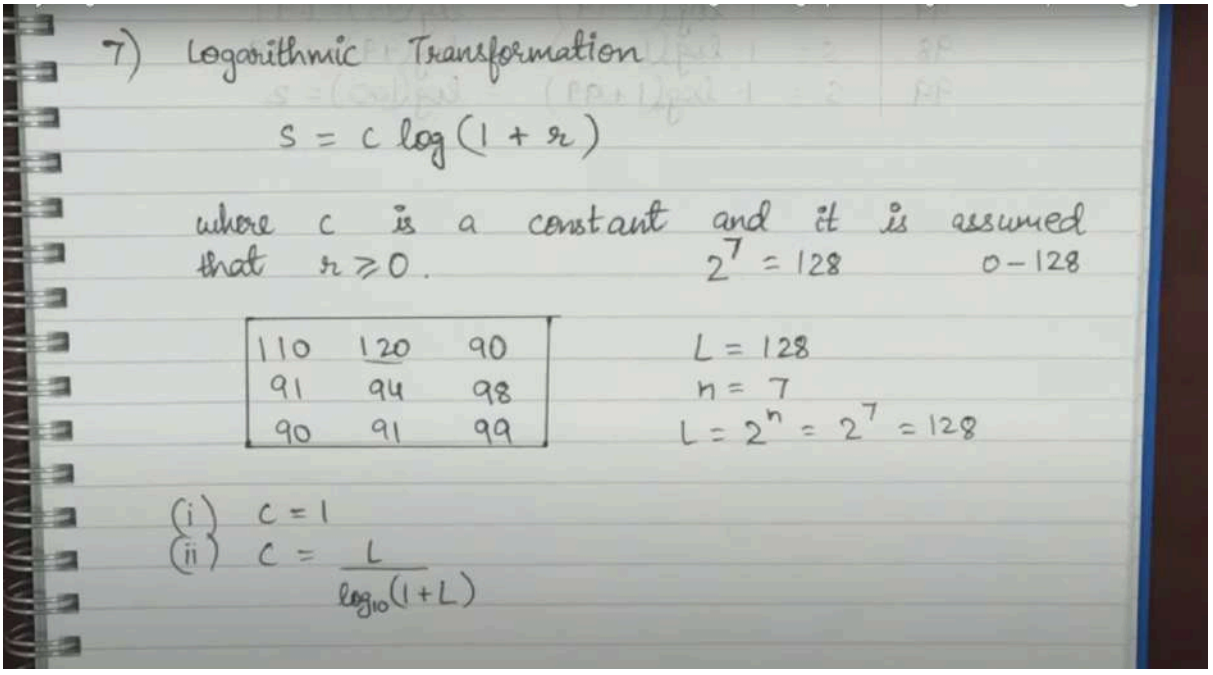

Q3. Apply laplacian filter on the given image.
Mask
50 x 8 − 50 x 8
= 0

So we get


Q3. Apply laplacian filter on the given image.
Mask
50 x 8 − 50 x 8
= 0
50 x 5 + 100 x 3 − 50 x 8
= 450 + 300 − 400
= 150.
50 x 3 + 100 x 5 − 400
= 150 + 500 − 400
= − 150.

Logarithmic Transformation and Power- law Transformation in digital image processing with examples





(i)

$c = 1$

r	s
110	$s = 1 \cdot \log_{10}(1+110) = \log(111) = 2.04 \approx 2$
120	$s = 1 \cdot \log_{10}(1+120) = \log(121) = 2.08 \approx 2$
90	$s = 1 \cdot \log_{10}(1+90) = \log(91) = 1.95 \approx 2$
91	$s = 1 \cdot \log_{10}(1+91) = \log(92) = 1.96 \approx 2$
94	$s = 1 \cdot \log_{10}(1+94) = \log(95) = 1.97 \approx 2$
98	$s = 1 \cdot \log_{10}(1+98) = \log(99) = 1.99 \approx 2$
99	$s = 1 \cdot \log_{10}(1+99) = \log(100) = 2$

$$s = c \log(1+r)$$
$$= \log(1+r)$$

91	$s = 1 \cdot \log_{10}(1+91) = \log(92) = 1.96 \approx 2$
94	$s = 1 \cdot \log_{10}(1+94) = \log(95) = 1.97 \approx 2$
98	$s = 1 \cdot \log_{10}(1+98) = \log(99) = 1.99 \approx 2$
99	$s = 1 \cdot \log_{10}(1+99) = \log(100) = 2$

$$s = c \log(1+r)$$
$$= \log(1+r)$$

110	120	90
91	94	98
90	91	99

Input image

→

2	2	2
2	2	2
2	2	2

Output image

94	$s = 1 \cdot \log_{10}(1+94) = \log(95) = 1.97 \approx 2$
98	$s = 1 \cdot \log_{10}(1+98) = \log(99) = 1.99 \approx 2$
99	$s = 1 \cdot \log_{10}(1+99) = \log(100) = 2$

$s = c \log(1+r)$
$= \log(1+r)$

110	120	90
91	94	98
90	91	99

Input image

→

2	2	2
2	2	2
2	2	2

Output image

(ii)

$L = 128$

$$c = \frac{L}{\log_{10}(1+L)} = \frac{128}{\log_{10}(1+128)} = \frac{128}{\log_{10}(129)} = 60.66$$

$s = c \log(1+r)$

$= \log(1+r)$

110	120	90
91	94	98
90	91	99

Input image

→

2	2	2
2	2	2
2	2	2

Output image

(ii)

$L = 128$

$$C = \frac{L}{\log_{10}(1+L)} = \frac{128}{\log_{10}(1+128)} = \frac{128}{\log_{10}(129)} = 60.66 \approx 61$$

$\approx 61.$

110	120	90
91	94	98
90	91	99

Input image

→

61 × 2.04	61 × 2.08	61 × 1.95
61 × 1.96	61 × 1.97	61 × 1.99
61 × 1.95	61 × 1.96	61 × 2

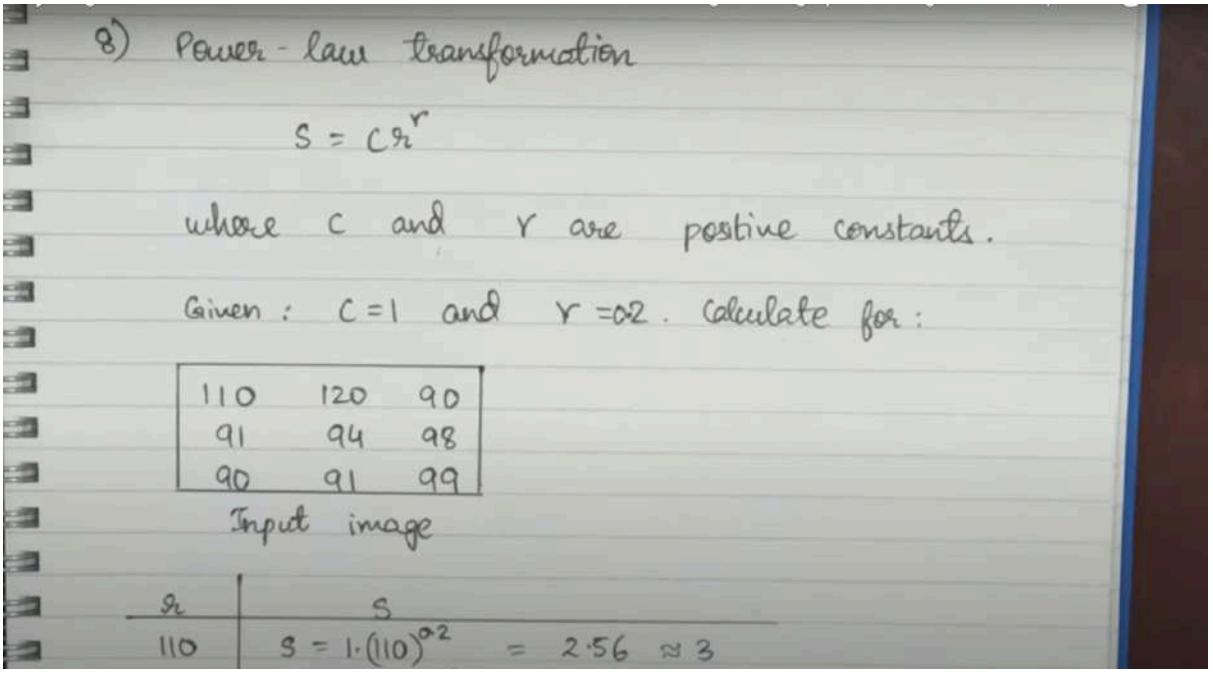
124.44	126.88	118.95
119.56	120.17	121.39
118.95	119.56	122

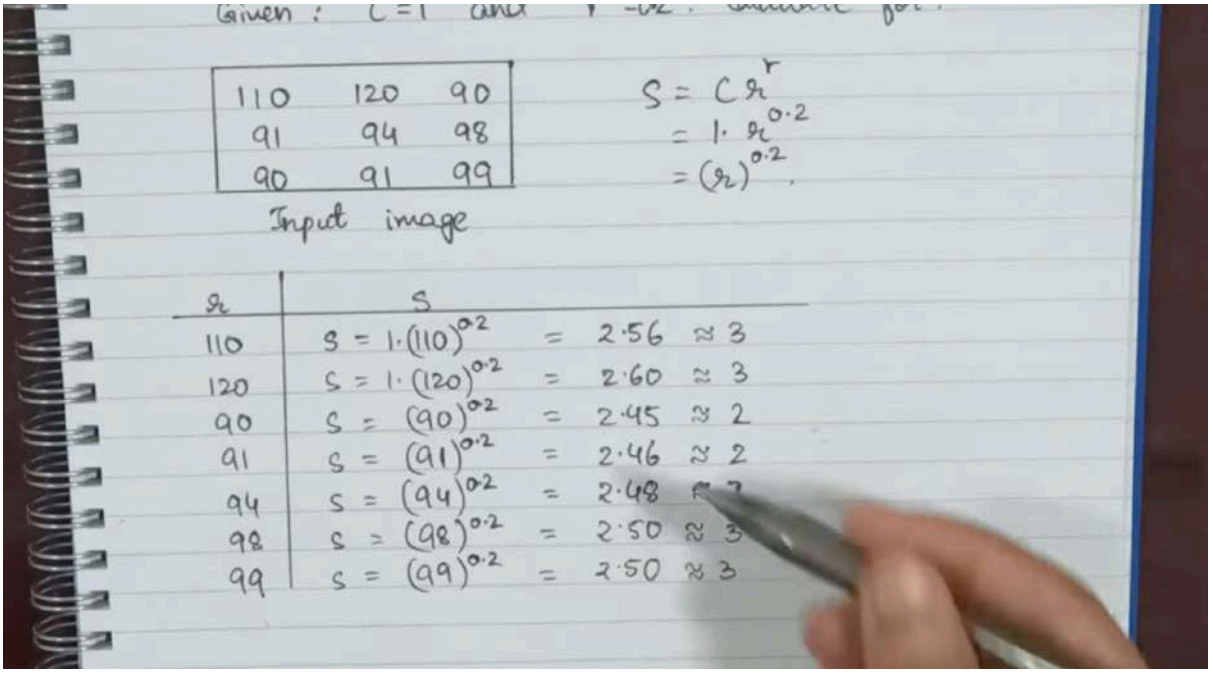
↓ Round off

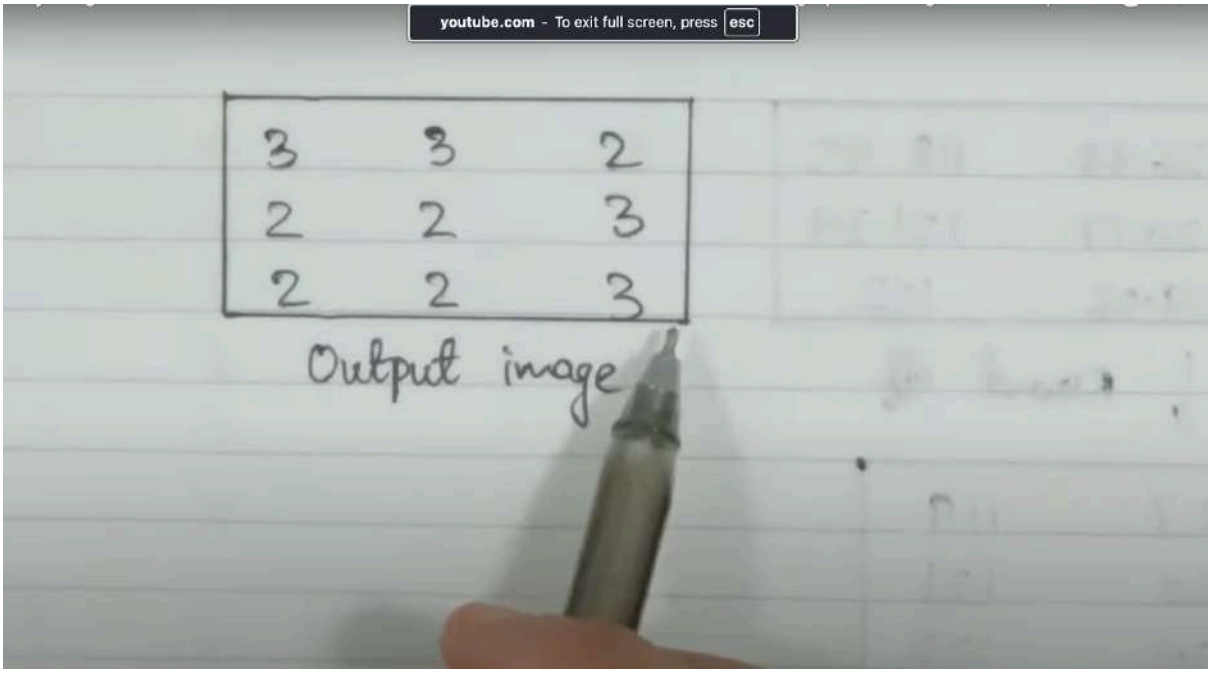
124	127	119
120	120	121
119	120	122

Output image

Power Law Transformation





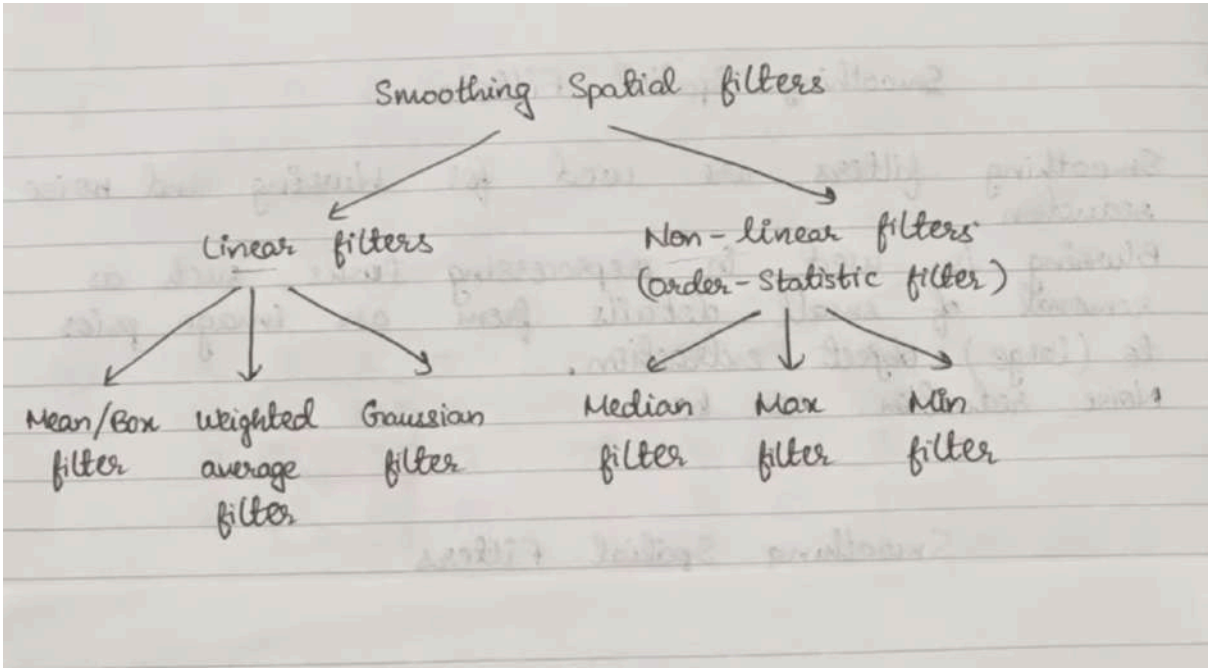


Smoothing Spatial Filters in digital image processing

→ Smoothing **filters** are used for **blurring** and **noise reduction**

→ **Blurring** is used in **preprocessing tasks** such as the **removal** of **small details** from an **image prior** to (**large**) **object extraction**.

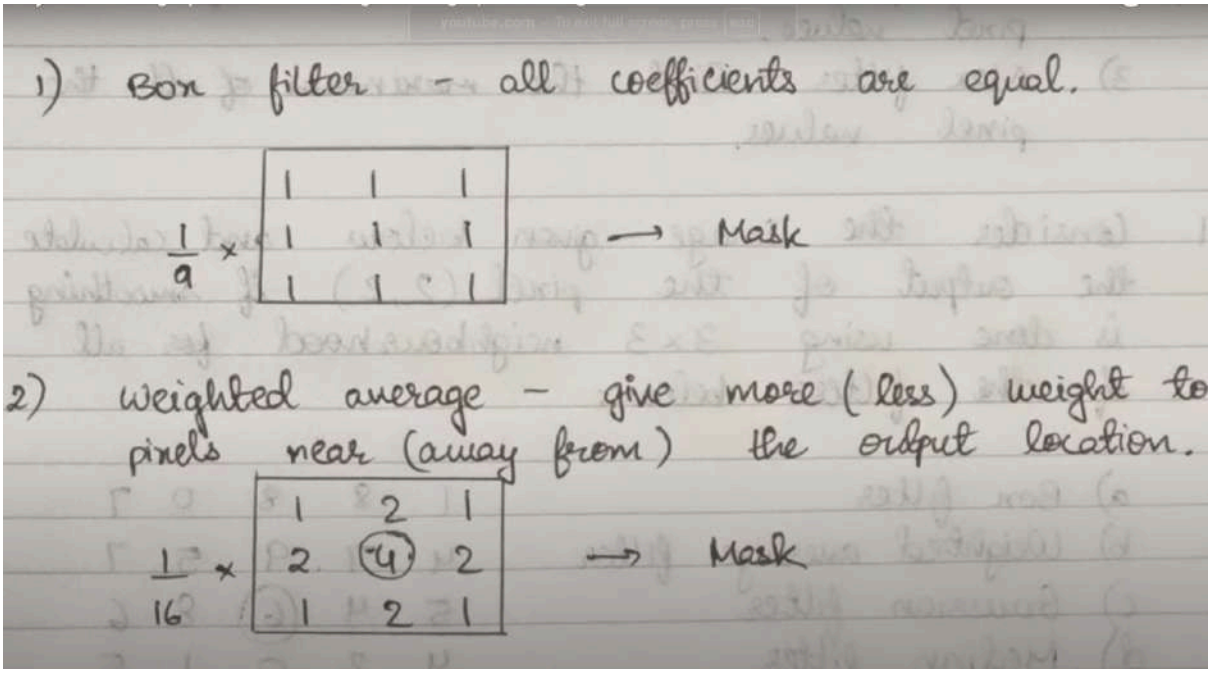
→ **Noise Reduction** can be **accomplished** by **blurring** with a **linear filter** and also by **non-linear filtering**.

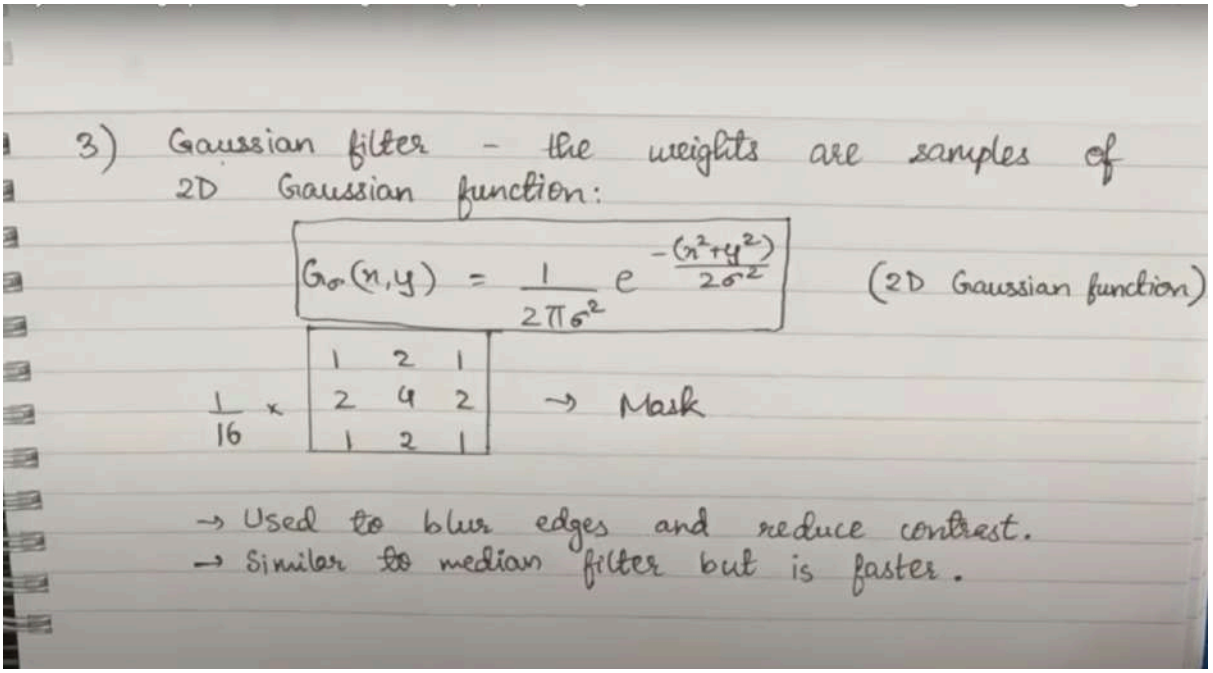


Smoothing Linear Filters

They are also known as averaging filters or lowpass filters as they are simply the average of the pixels contained in the neighbourhood of the filter mask.

The process results in an image with reduced sharp transitions in intensities which ultimately leads to noise reduction.

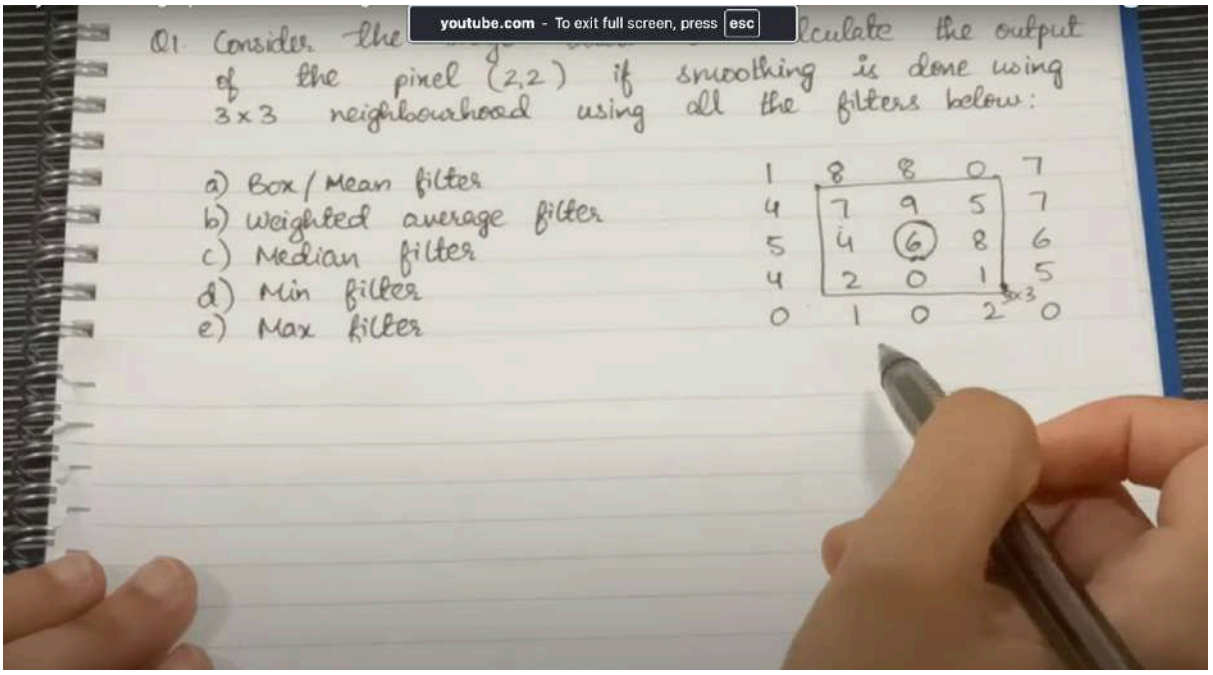


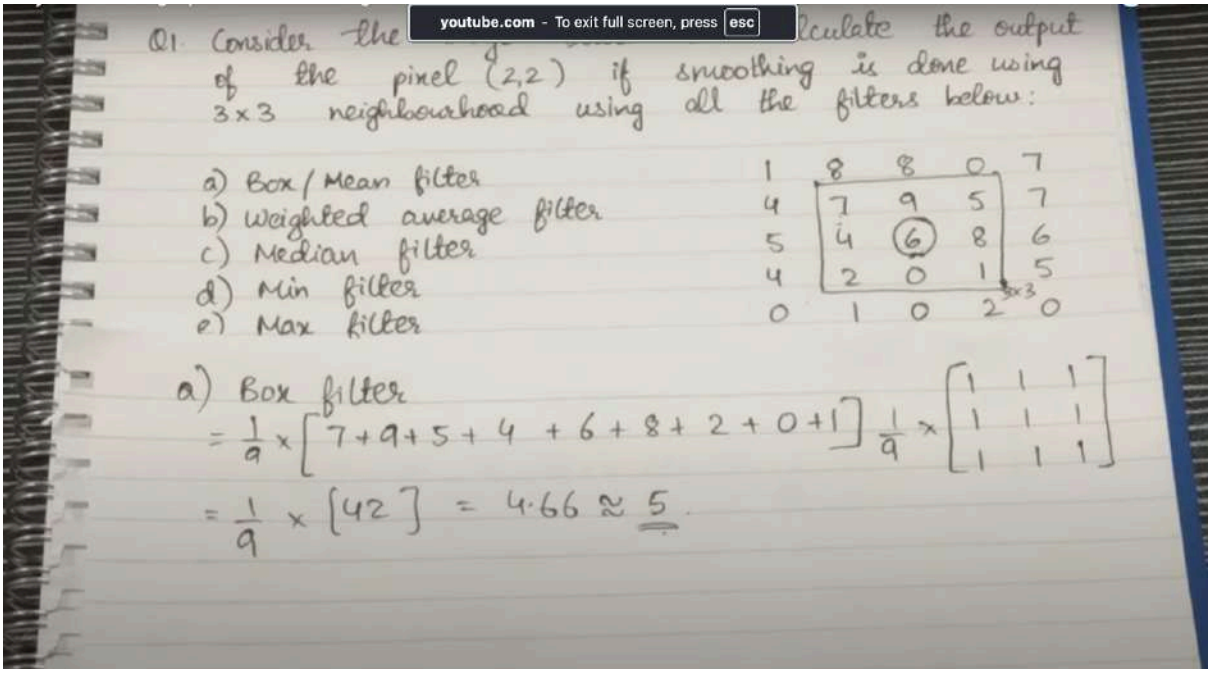


Non-linear (Order-Statistic) Filters

Their response is based on ordering(ranking) the pixels contained in the image area encompassed by the filter, and then replacing the value of the center pixel with the value determined by the ranking result.

1) **Median Filter** - **median** of the all pixels value
2) **Min Filter** - find the **minimum** pixel values **among** all the pixels
3) **Max Filter** - find the **maximum** pixel values **among** all the pixels





Q1. Consider the image below and calculate the output of the pixel (2,2) if smoothing is done using 3×3 neighbourhood using all the filters below:

a) Box / Mean filter
b) Weighted average filter
c) Median filter
d) Min filter
e) Max filter

1	8	8	0	7
4	7	9	5	7
5	4	(6)	8	6
4	2	0	1	5
0	1	0	2	0

3×3

Input image

b) Weighted average filter

$$= \frac{1}{16}\left[\begin{matrix} 7\times1 + 9\times2 + 5\times1 + 4\times2 + \\ 4\times6 + 8\times2 + 2\times1 + 0\times2 \\ + 1\times1 \end{matrix}\right]$$

$$= \frac{1}{16}\left[81\right] = 5.0625 \approx \underline{5}.$$

$$\frac{1}{16}\times\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$

Q1. Consider the image below and calculate the output of the pixel (2,2) if smoothing is done using 3×3 neighbourhood using all the filters below:

a) Box / Mean filter
b) Weighted average filter
c) Median filter
d) Min filter
e) Max filter

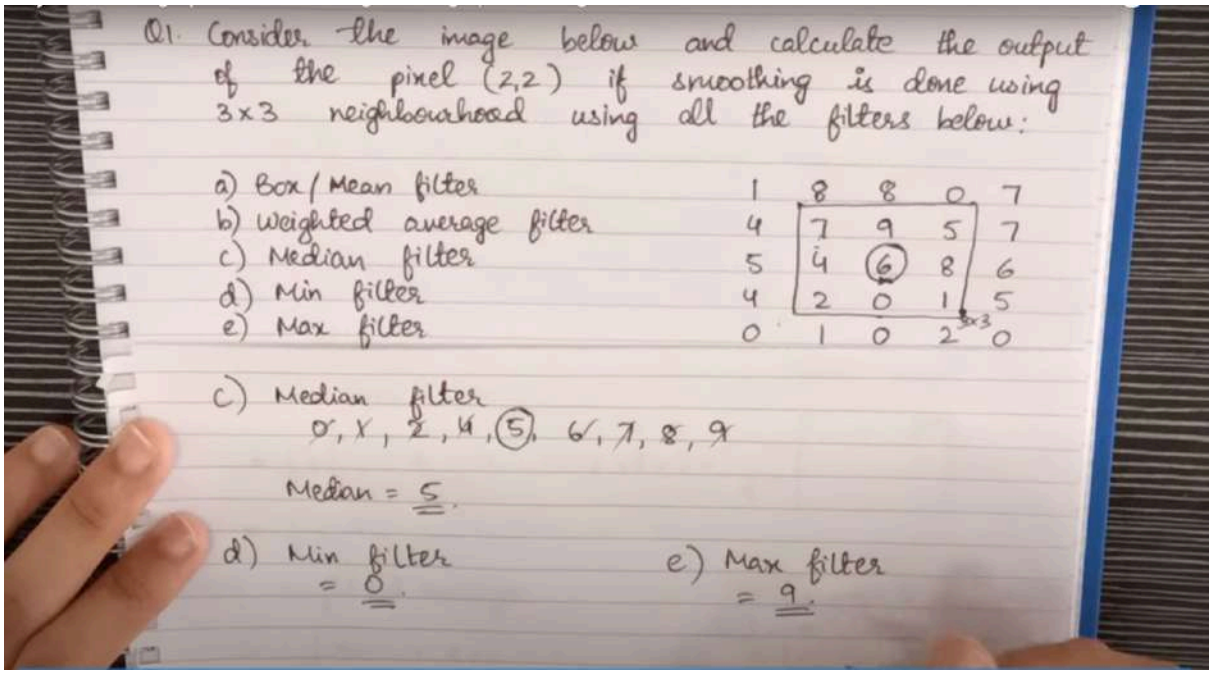
1	8	8	0	7
4	7	9	5	7
5	4	(6)	8	6
4	2	0	1	5
0	1	0	2	0

3×3

c) Median filter

0, 1, 2, 4, (5), 6, 7, 8, 9

Median = $\underline{5}$.



Histogram Matching and Equalization

2) Histogram Matching (Specification)

Q1. Given below are two histograms (i) and (ii). Modify the histogram (i) as given by histogram (ii).

(i)

Gray level(r_k)	0	1	2	3	4	5	6	7
No. of pixels(n_k)	80	100	90	60	30	20	10	0

(ii)

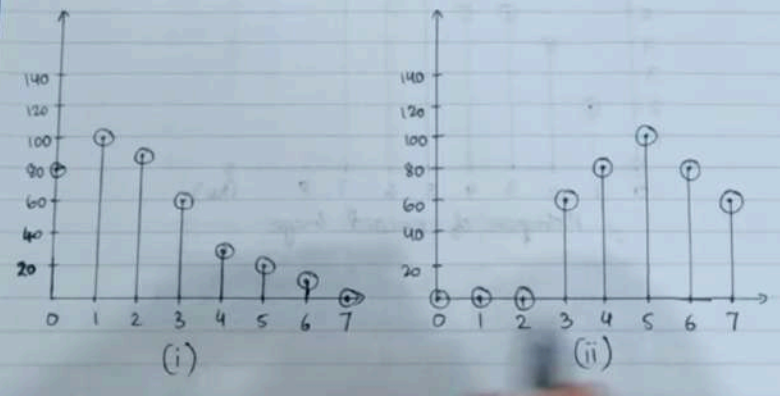
Gray level(r_k)	0	1	2	3	4	5	6	7
No. of pixels(n_k)	0	0	0	60	80	100	80	70

(i)

Gray level(r_k)	0	1	2	3	4	5	6	7
No. of pixels(n_k)	80	100	90	60	30	20	10	0

(ii)

Gray level(r_k)	0	1	2	3	4	5	6	7
No. of pixels(n_k)	0	0	0	60	80	100	80	70

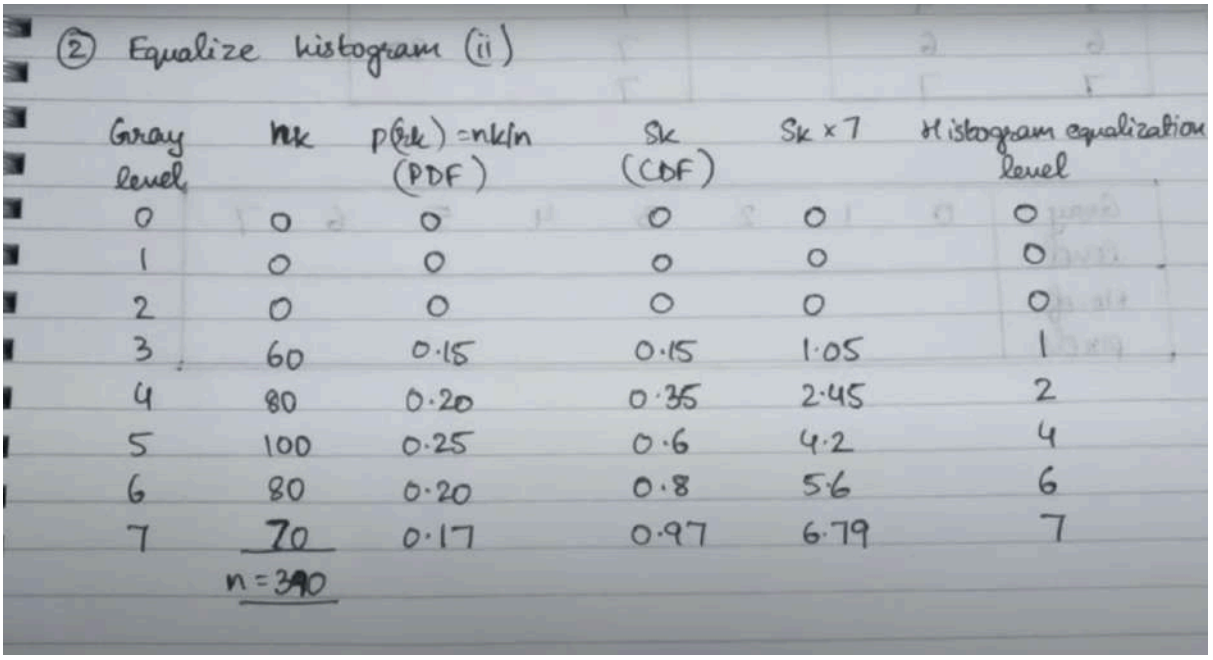


① Equalize histogram(i)

Gray level	n_k	$p(r_k) = n_k/n$ (PDF)	S_k (CDF)	$S_k \times 7$	Histogram equalization	New n_k
0	80	0.20	0.20	1.4	1	80
1	100	0.25	0.45	3.15	3	100
2	90	0.23	0.68	4.76	5	90
3	60	0.15	0.83	5.81	6	60+30 = 90
4	30	0.07	0.9	6.3	6	
5	20	0.05	0.95	6.65	7	
6	10	0.02	0.97	6.79	7	20+10 = 30
7	0	0	0.97	6.79	7	
	$n = 390$					

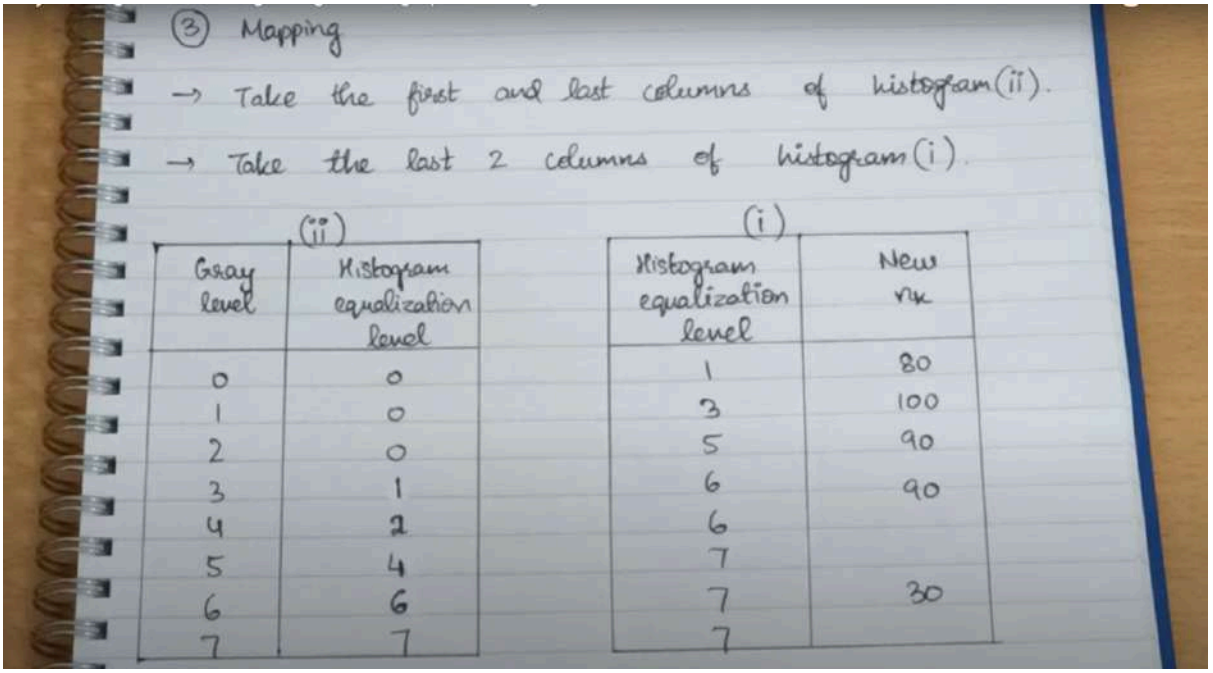
① Equalize histogram(i)

Gray level	n_k	$p(r_k) = n_k/n$ (PDF)	S_k (CDF)	$S_k \times 7$	Histogram equalization	New n_k
0	80	0.20	0.20	1.4	1	80
1	100	0.25	0.45	3.15	3	100
2	90	0.23	0.68	4.76	5	90
3	60	0.15	0.83	5.81	6	60+30 = 90
4	30	0.07	0.9	6.3	6	
5	20	0.05	0.95	6.65	7	
6	10	0.02	0.97	6.79	7	20+10 = 30
7	0	0	0.97	6.79	7	
	$n = 390$					



② Equalize histogram (ii)

Gray level	nk	p(rk) = nk/n (PDF)	Sk (CDF)	Sk × 7	Histogram equalization level
0	0	0	0	0	0
1	0	0	0	0	0
2	0	0	0	0	0
3	60	0.15	0.15	1.05	1
4	80	0.20	0.35	2.45	2
5	100	0.25	0.6	4.2	4
6	80	0.20	0.8	5.6	6
7	70	0.17	0.97	6.79	7
	n = 390				



③ Mapping

→ Take the first and last columns of histogram (ii).

→ Take the last 2 columns of histogram (i).

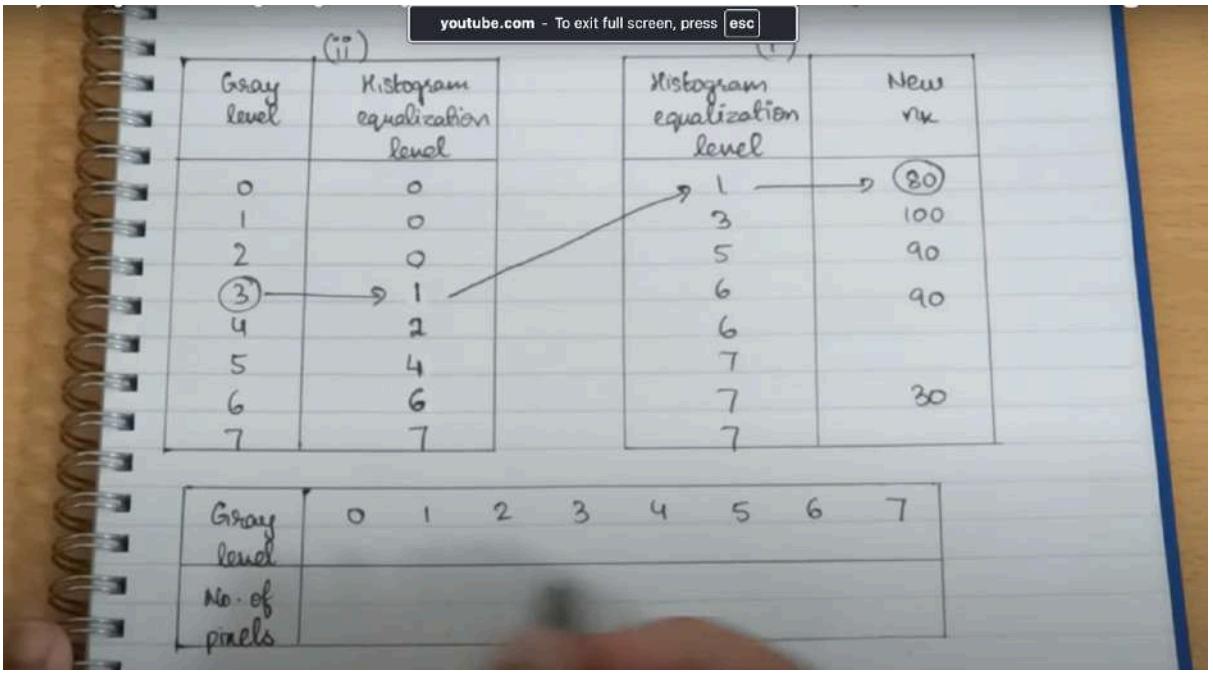
(ii)

Gray level	Histogram equalization level
0	0
1	0
2	0
3	1
4	2
5	4
6	6
7	7

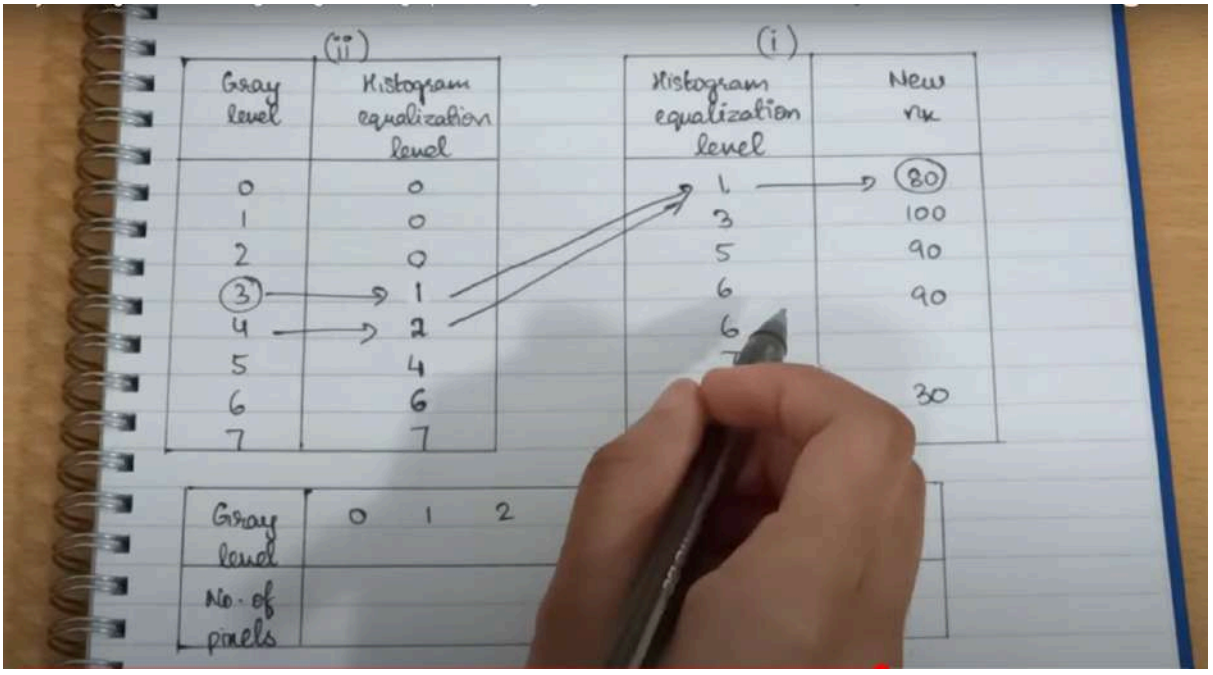
(i)

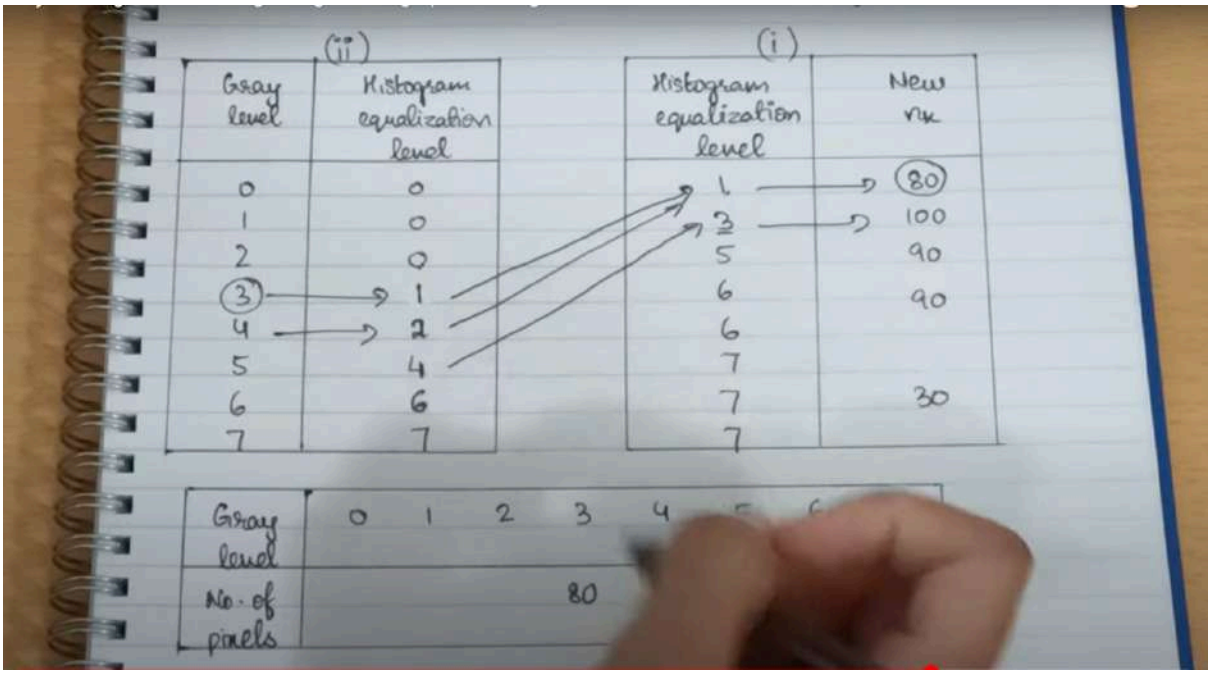
Histogram equalization level	New nk
1	80
3	100
5	90
6	90
6	
7	
7	30
7	

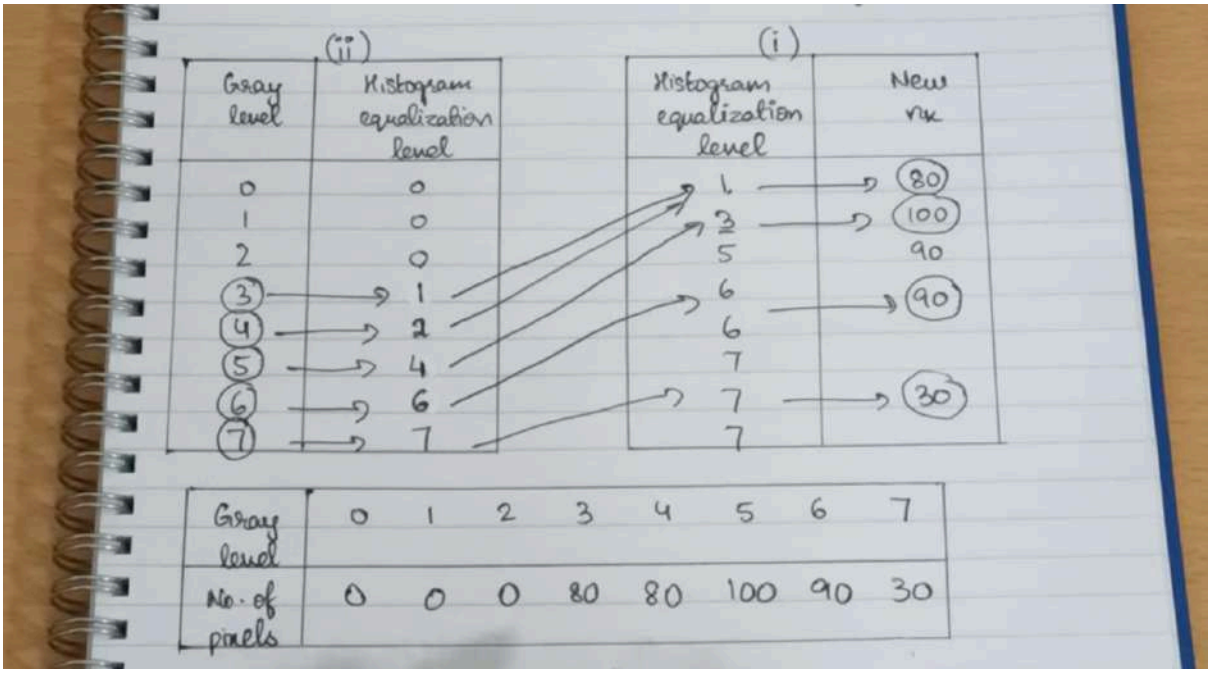
In table ii we have 3 → 1 so if we find the 1 we get in table ii as

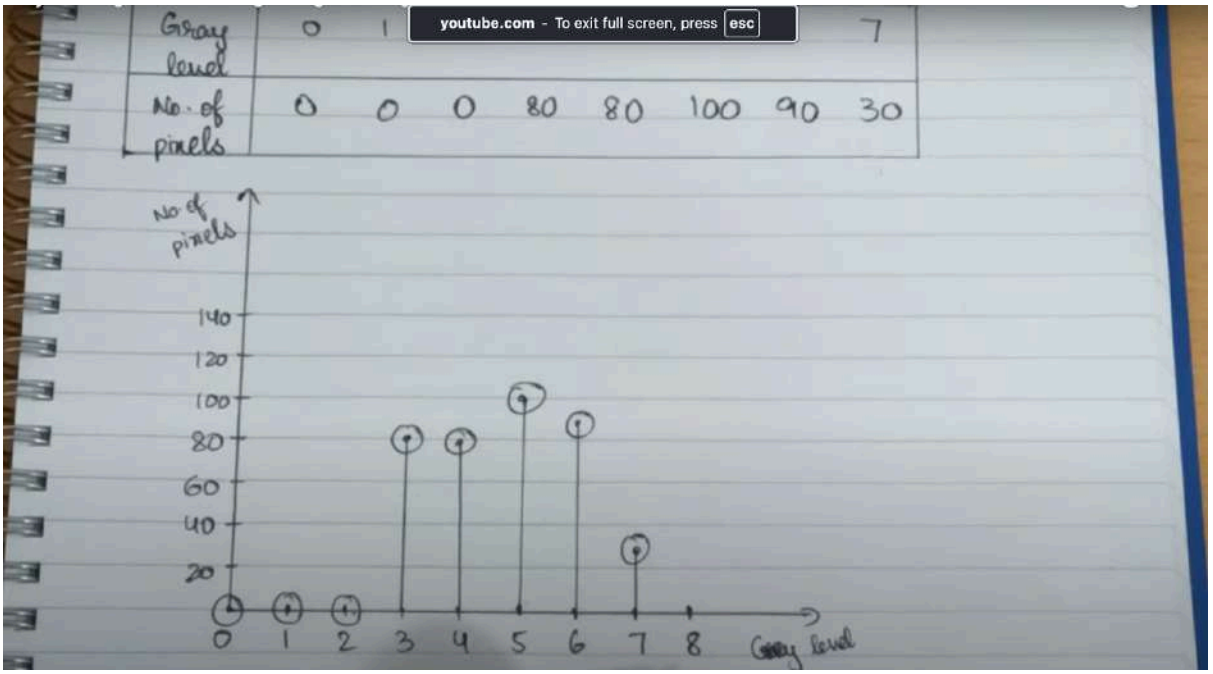


Now we move to 4 and 4 has 2 that means the nearest are 3 and 1 but we must select less than 2 so it will be 1




(ii)
Gray level
Histogram equalization level
0 0
1 0
2 0
3 1
4 2
5 4
6 6
7 7
(i)
Histogram equalization level
New rk
1 → 80
3 → 100
5 → 90
6 → 90
6
7
7 → 30
7
Gray level
0 1 2 3 4
No. of pixels
80


(ii)
Gray level
Histogram equalization level
0 0
1 0
2 0
3 → 1
4 → 2
5 → 4
6 → 6
7 → 7
(i)
Histogram equalization level
New rk
1 → 80
3 → 100
5 → 90
6 → 90
6
7
7 → 30
7
Gray level
0 1 2 3 4 5 6 7
No. of pixels
0 0 0 80 80 100 90 30



Fundamentals of Spatial Filtering in digital image processing

→ The name filter is borrowed from frequency domain processing. It refers to accepting (passing). It refers to accepting (passing) or rejecting certain frequency components.
Eg: A filter that passes low frequencies is called a lowpass filter.
We can accomplish a similar smoothing directly on the image itself using spatial filters (also called masks, kernels, templates, and windows).

1. **Convolution**

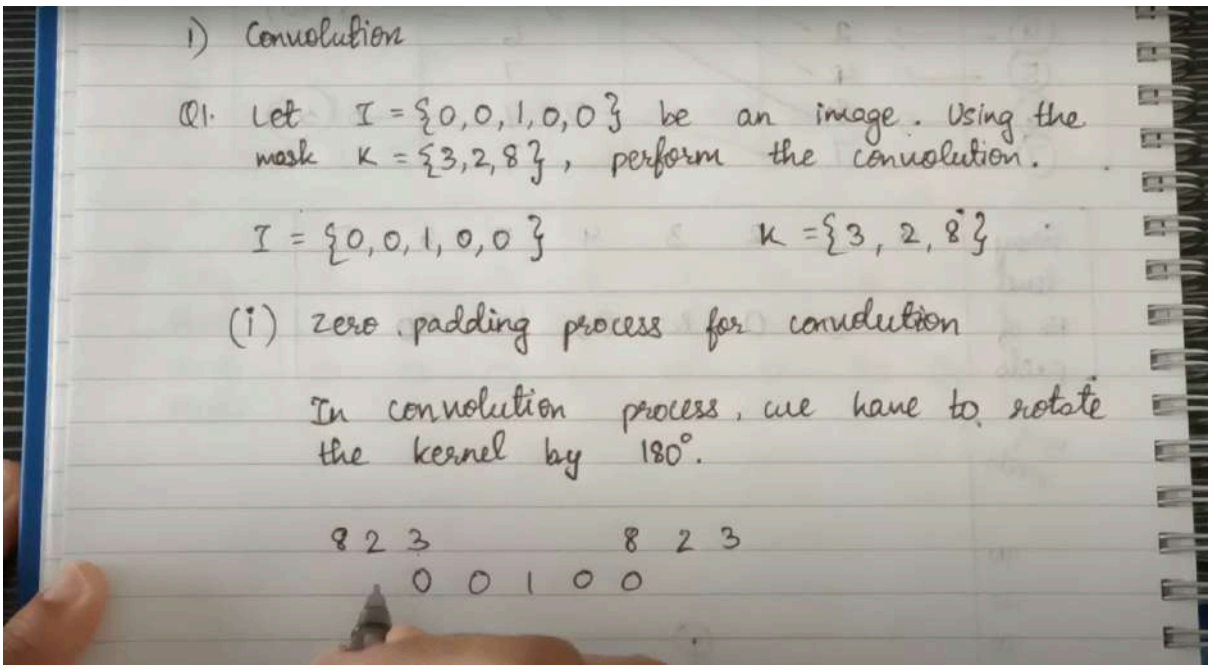
1) Convolution

Q1. Let $I = \{0, 0, 1, 0, 0\}$ be an image. Using the mask $K = \{3, 2, 8\}$, perform the convolution.

$I = \{0, 0, 1, 0, 0\}$ $\quad\quad$ $K = \{3, 2, 8\}$

(i) zero padding process for convolution

In convolution process, we have to rotate the kernel by 180°.



1) Convolution

Q1. Let $I = \{0, 0, 1, 0, 0\}$ be an image. Using the mask $K = \{3, 2, 8\}$, perform the convolution.

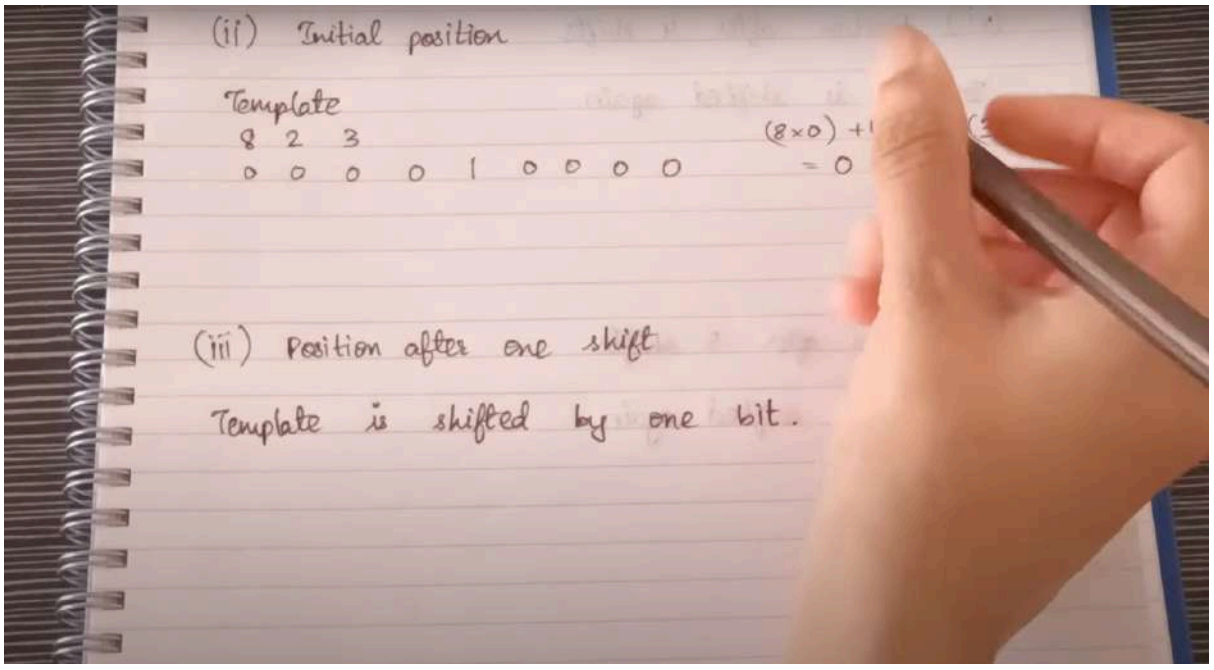
$I = \{0, 0, 1, 0, 0\}$ $\quad\quad$ $K = \{3, 2, 8\}$

(i) zero padding process for convolution

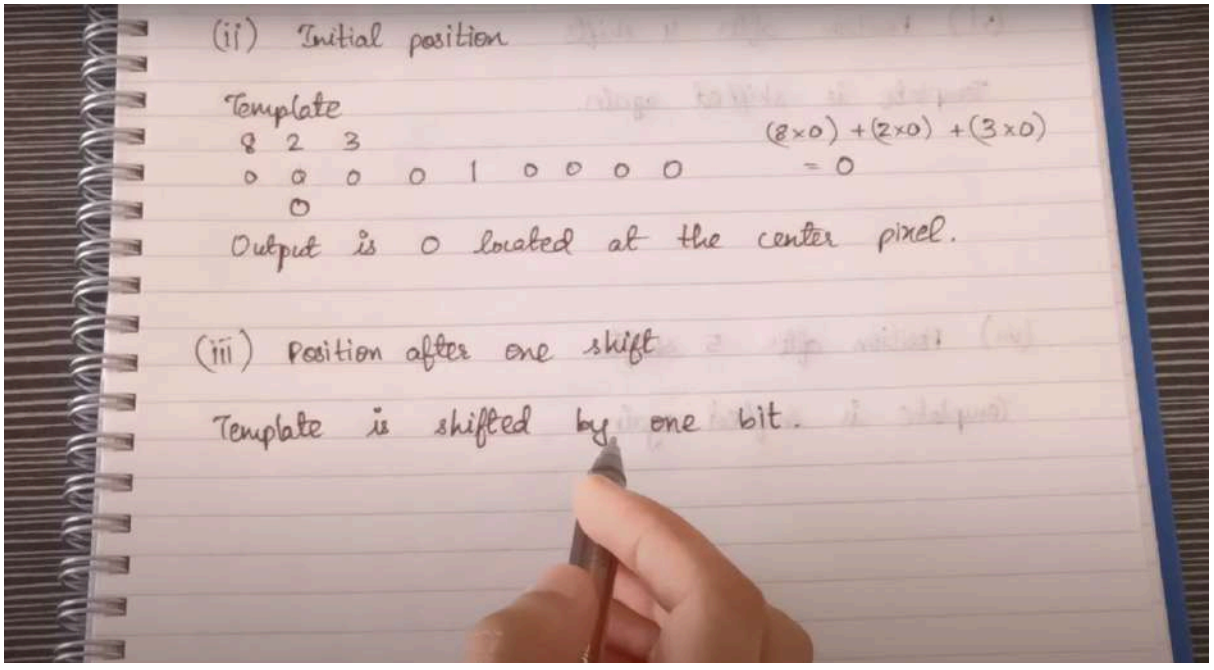
In convolution process, we have to rotate the kernel by 180°.

8 2 3 $\quad\quad$ 8 2 3

0 0 1 0 0



Now we calculated the 0 where to keep this 0? The answer is at the mid-position



(ii) Initial position

Template

8 2 3

0 0 0 0 1 0 0 0 0

0

$(8\times0)+(2\times0)+(3\times0)$

$=0$

Output is 0 located at the center pixel.

(iii) Position after one shift

Template is shifted by one bit.

8 2 3

0 0 0 0 1 0 0 0 0

0

(ii) Initial position

Template

8 2 3

0 0 0 0 1 0 0 0 0

0

$(8\times0)+(2\times0)+(3\times0)$

$=0$

Output is 0 located at the center pixel.

(iii) Position after one shift

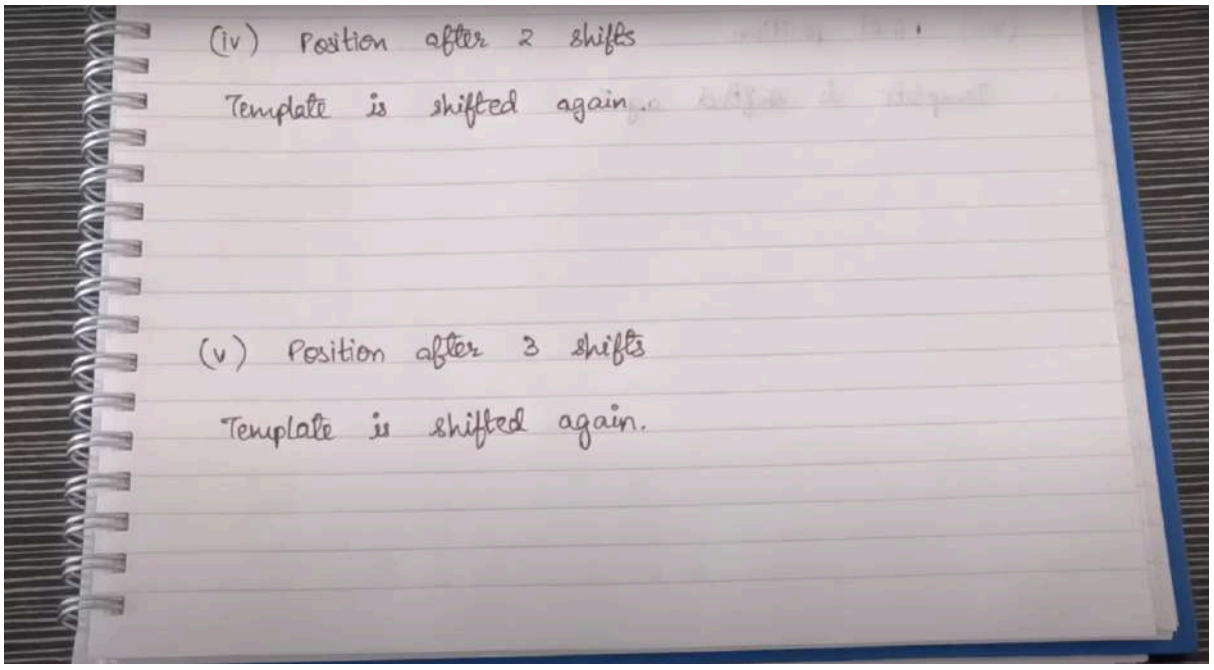
Template is shifted by one bit.

8 2 3

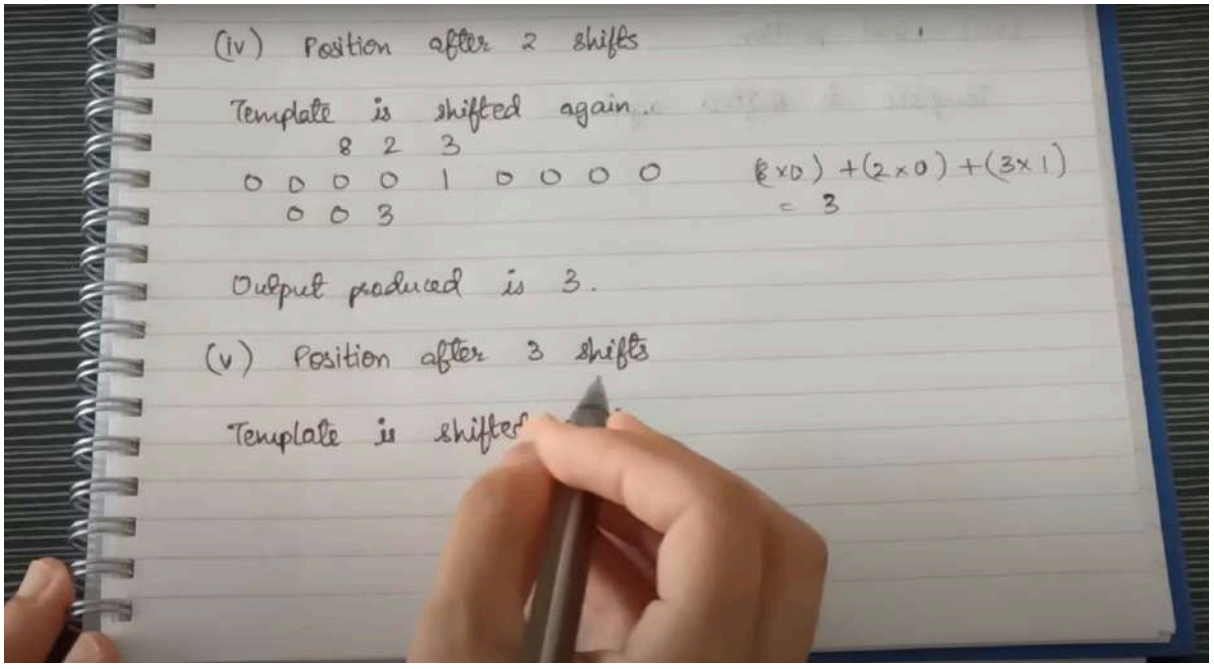
0 0 0 0 1 0 0 0 0

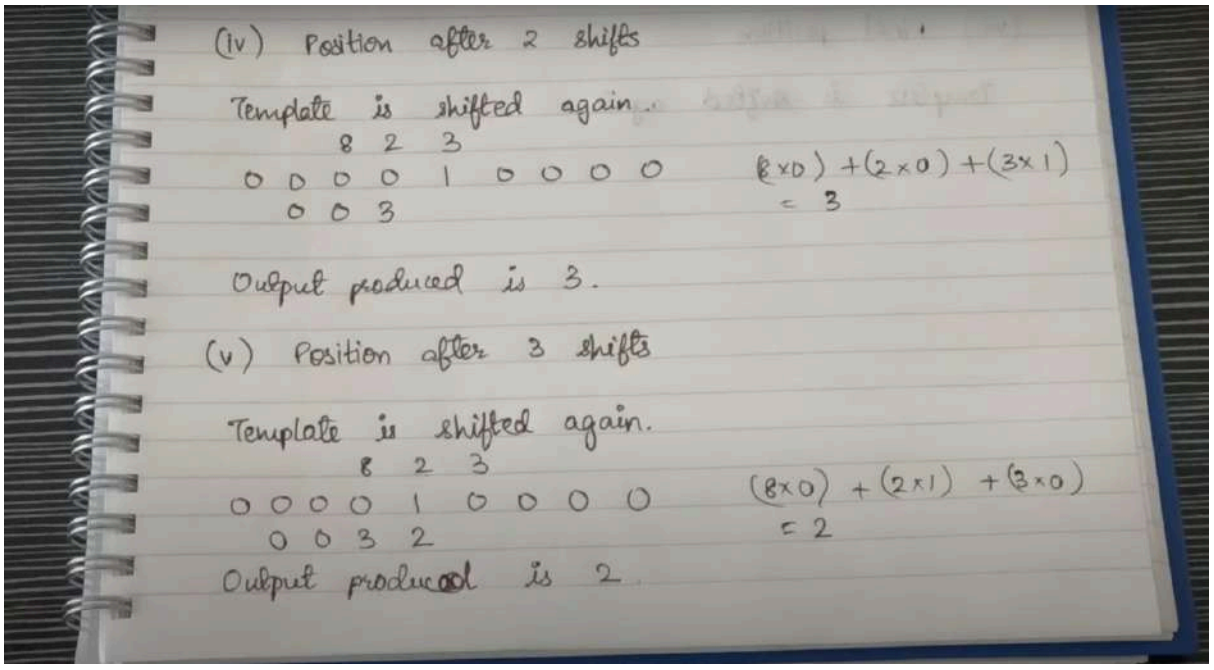
0 0

Output is 0.

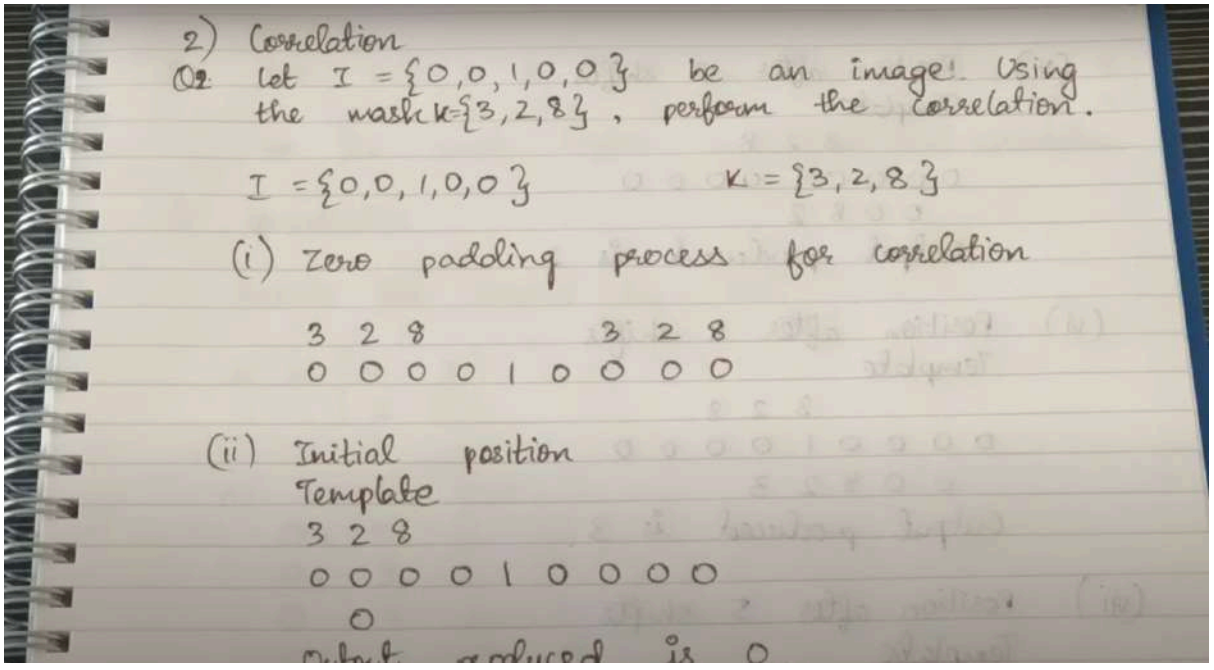


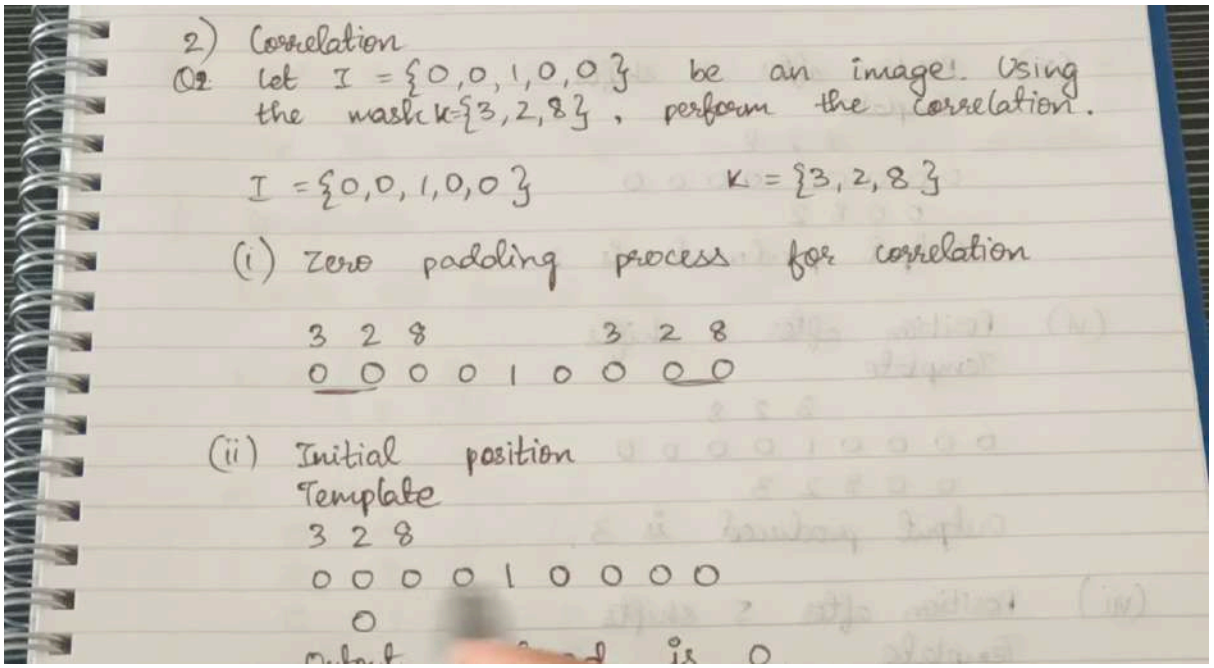
After 2 shifts we get value





The main difference between Convolutional and Correlation is that in Convt we rotate by 180 but in Correlation, we do not rotate but keep it as it is





2) Correlation

Q2 Let I = {0,0,1,0,0} be an image. Using the mask K={3,2,8}, perform the correlation.

I = {0,0,1,0,0} K = {3,2,8}

(i) Zero padding process for correlation

3 2 8 3 2 8

0 0 0 0 1 0 0 0 0

(ii) Initial position

Template

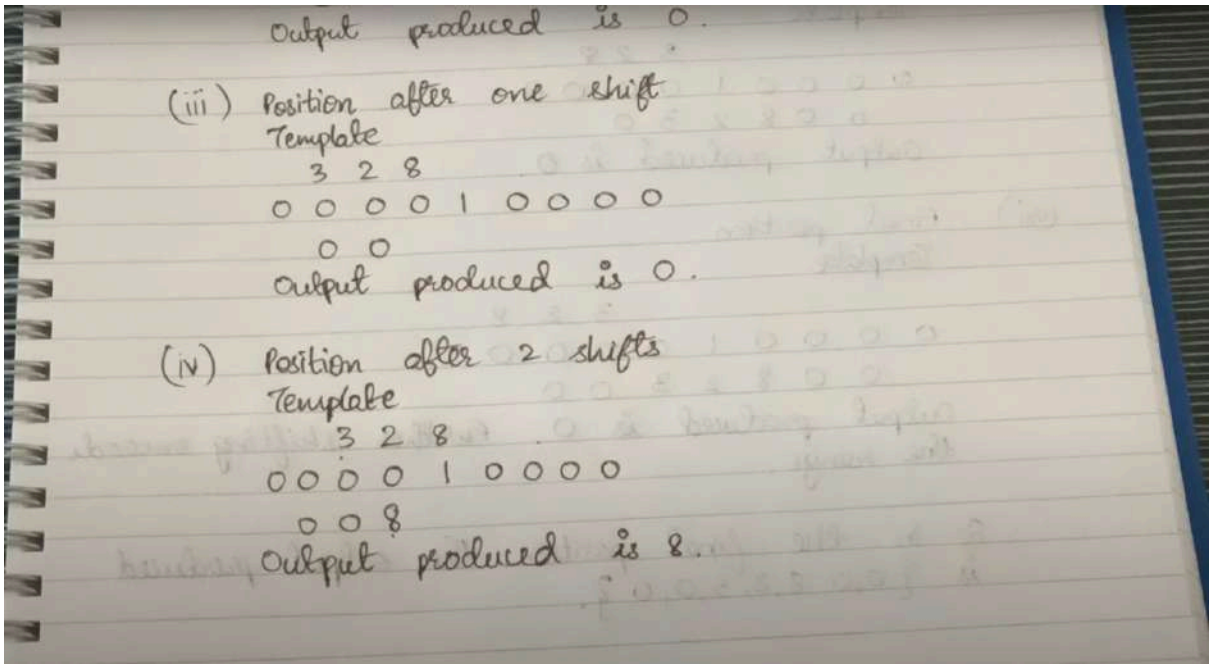
3 2 8

0 0 0 0 1 0 0 0 0

0

Output produced is 0

We only pad the first and last two zeros



Output produced is 0.

(iii) Position after one shift

Template

3 2 8

0 0 0 0 1 0 0 0 0

0 0

Output produced is 0.

(iv) Position after 2 shifts

Template

3 2 8

0 0 0 0 1 0 0 0 0

0 0 8

Output produced is 8.

(v) Position after 3 shifts
Template

```
      3 2 8
0 0 0 0 1 0 0 0 0
  0 0 8 2
```

Output produced is 2.

(vi) Position after 4 shifts
Template

```
        3 2 8
0 0 0 0 1 0 0 0 0
  0 0 8 2 3
```

Output produced is 3.

(vii) Position after 5 shifts
Template

```
          3 2 8
0 0 0 0 1 0 0 0 0
  0 0 8 2 3 0
```

Output produced is 0.

(viii) Final position
Template

```
            3 2 8
0 0 0 0 1 0 0 0 0
  0 0 8 2 3 0 0
```

Output produced is 0. Further shifting exceeds the range.

So in the final position, the output produced is {0,0,8,2,3,0,0}.

Q3. Let $I = \begin{pmatrix} 3 & 3 \\ 3 & 3 \end{pmatrix}$ be an image and $k = \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}$ be a kernel (mask). Perform convolution and correlation.

i) Convolution

Rotate the kernel by 180°.

$k' =$

1 2
3 4

4
3 – 4
1 – 2
1

a)

0	0	0	0
0	3	3	0
0	3	3	0
0	0	0	0

$k' = \begin{pmatrix} 4 & 3 \\ 2 & 1 \end{pmatrix}$

a)

$$\begin{matrix} 0_4 & 0_3 & 0 & 0 \\ 0_2 & 3_1 & 3 & 0 \\ 0 & 3 & 3 & 0 \\ 0 & 0 & 0 & 0 \end{matrix} \rightarrow \begin{matrix} 3 & 0 & 0 & 0 \\ 0 & 3 & 3 & 0 \\ 0 & 3 & 3 & 0 \\ 0 & 0 & 0 & 0 \end{matrix}$$

$$K' = \begin{pmatrix} 4 & 3 \\ 2 & 1 \end{pmatrix}$$

b)

$$\begin{matrix} 3 & 0_4 & 0_3 & 0 \\ 0 & 3_2 & 3_1 & 0 \\ 0 & 3 & 3 & 0 \\ 0 & 0 & 0 & 0 \end{matrix} \rightarrow \begin{matrix} 3 & 9 & 0 & 0 \\ 0 & 3 & 3 & 0 \\ 0 & 3 & 3 & 0 \\ 0 & 0 & 0 & 0 \end{matrix}$$

$(4 \times 0) + (3 \times 0) + (2 \times 3) + (1 \times 3)$

$= 0 + 0 + 6 + 3 = 9.$

d)

$$\begin{matrix} 3 & 9 & 6 & 0 \\ 0_4 & 3_3 & 3 & 0 \\ 0_2 & 3_1 & 3 & 0 \\ 0 & 0 & 0 & 0 \end{matrix} \rightarrow \begin{matrix} 3 & 9 & 6 & 0 \\ 12 & 3 & 3 & 0 \\ 0 & 3 & 3 & 0 \\ 0 & 0 & 0 & 0 \end{matrix}$$

e)

$$\begin{matrix} 3 & 9 & 6 & 0 \\ 12 & 3_4 & 3_3 & 0 \\ 0 & 3_2 & 3_1 & 0 \\ 0 & 0 & 0 & 0 \end{matrix} \rightarrow \begin{matrix} 3 & 9 & 6 & 0 \\ 12 & 30 & 3 & 0 \\ 0 & 3 & 3 & 0 \\ 0 & 0 & 0 & 0 \end{matrix}$$

$(4 \times 3) + (3 \times 3) + (2 \times 3) + (1 \times 3)$

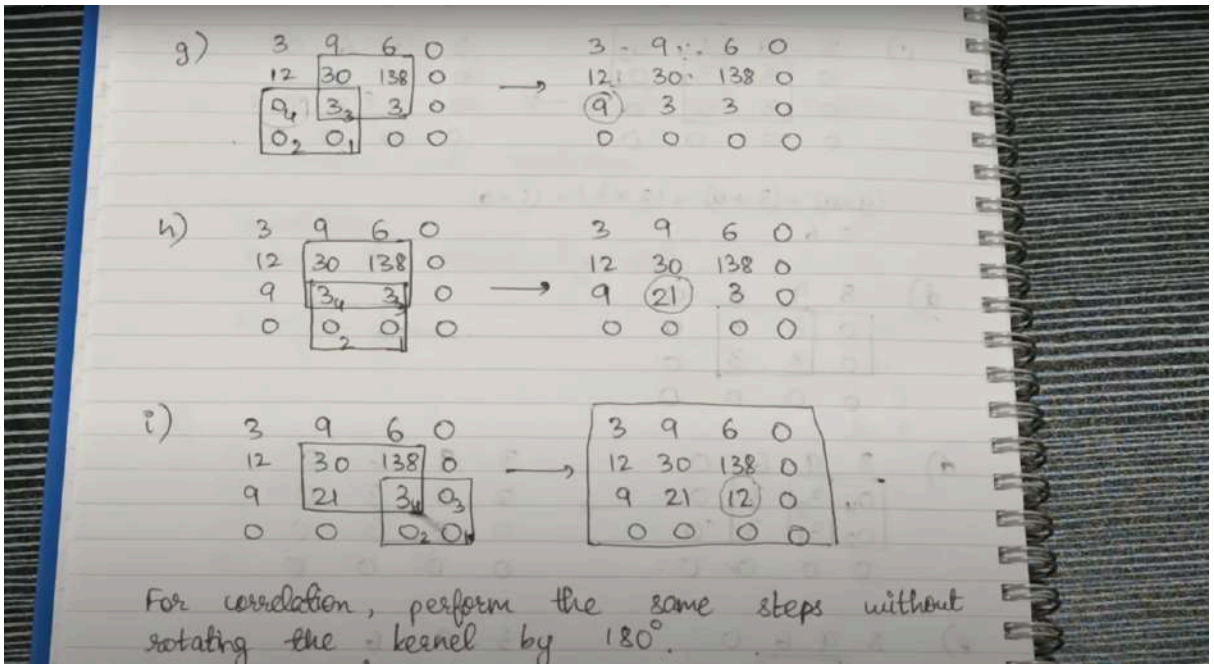
$= 12 + 9 + 6 + 3$

$= 30.$

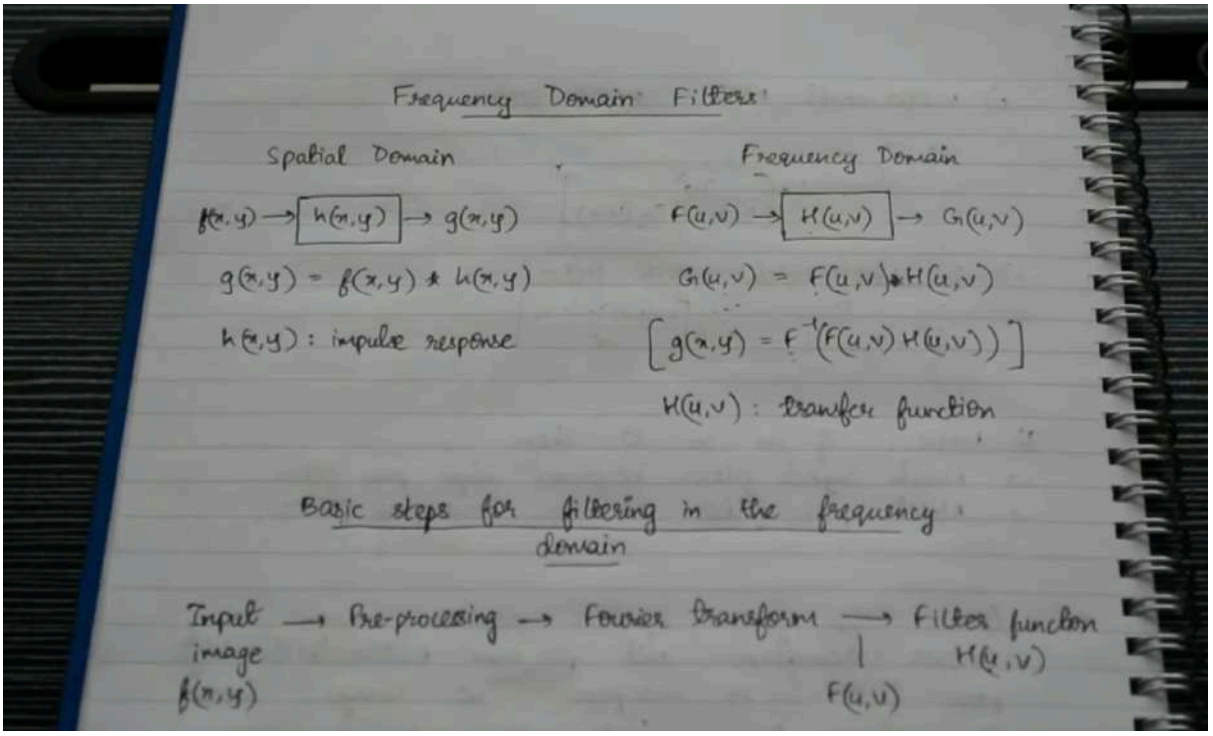
f)

$$\begin{matrix} 3 & 9 & 6 & 0 \\ 12 & 30_4 & 3_3 & 0 \\ 0 & 3_2 & 3_1 & 0 \\ 0 & 0 & 0 & 0 \end{matrix} \rightarrow \begin{matrix} 3 & 9 & 6 & 0 \\ 12 & 30 & 138 & 0 \\ 0 & 3 & 3 & 0 \\ 0 & 0 & 0 & 0 \end{matrix}$$

$120 + 9 + 6 + 3 = 138.$



Frequency domain filtering in image processing - Low pass and High pass filters



In Spatial Domain

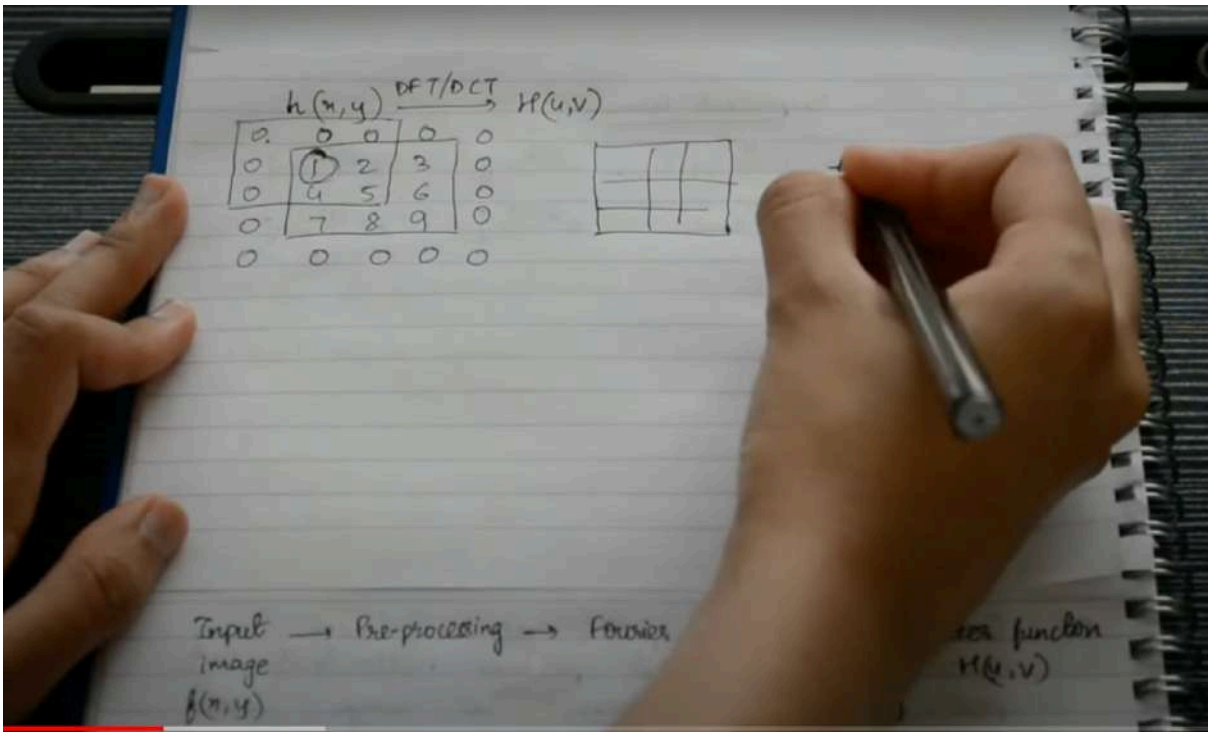
1. We do padding so that edges are not lost for example

   1 2 3
   3 4 5
   3 4 2
   Then after padding this will be

   0 0 0 0 0
   0 1 2 3 0
   0 3 4 5 0
   0 3 4 2 0
   0 0 0 0 0
   Suppose we have a mask kernel filter matrix

   0 1 0
   0 0 1
   1 0 0

   Then slide the mask over the padded matrix multiply each pixel sum them and replace the old pixel with the new one



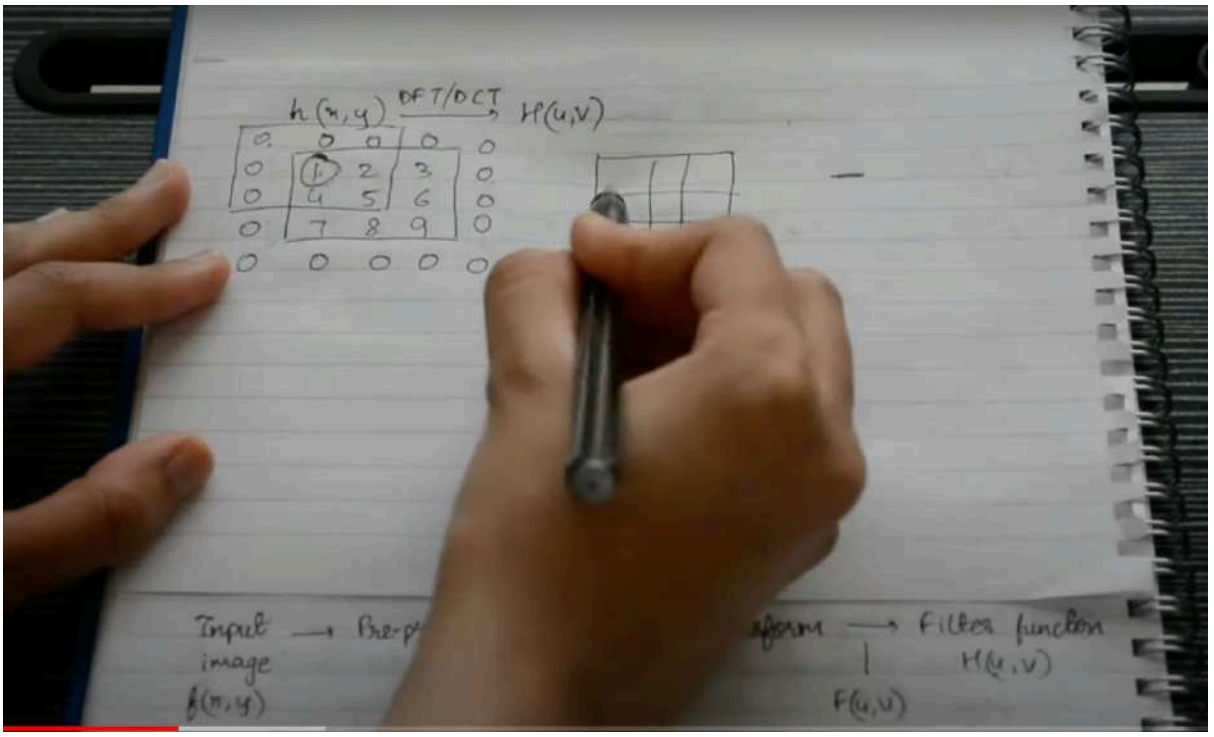
But in the frequency domain, we do pixel-by-pixel multiplication images.

1 2 3
3 4 5
3 4 2
Then after padding this will be

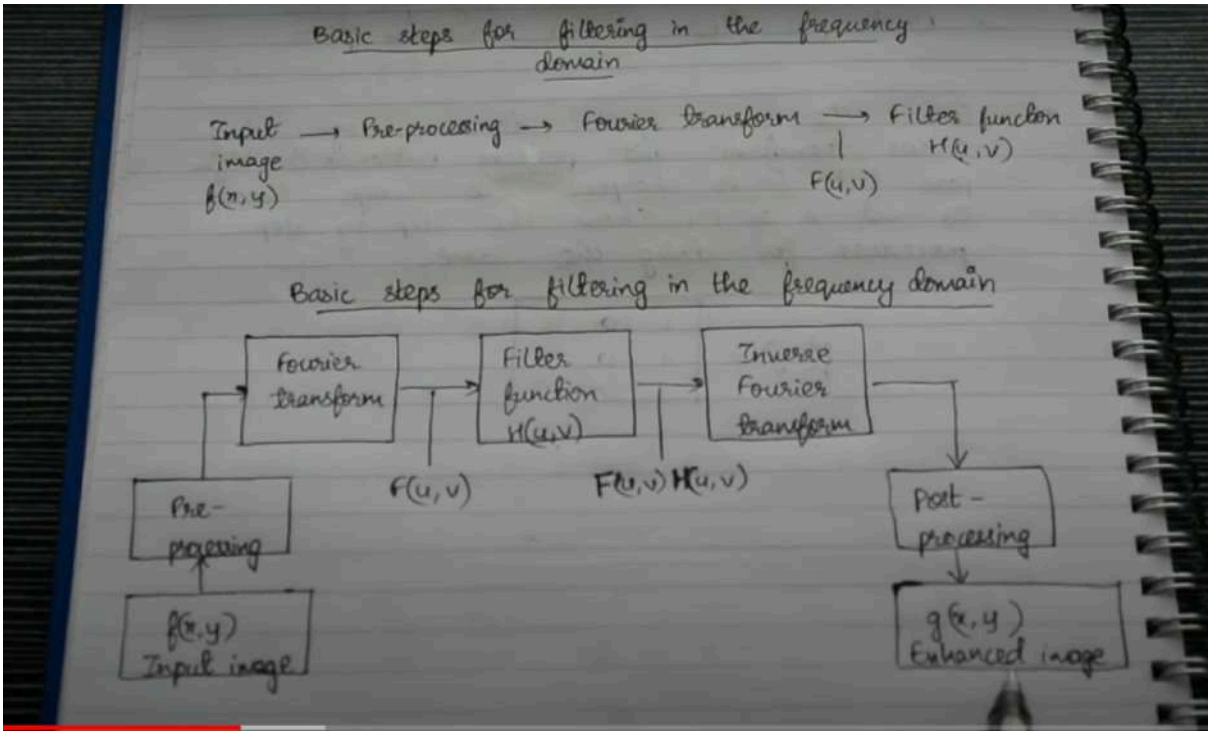
0 0 0 0 0
0 1 2 3 0
0 3 4 5 0
0 3 4 2 0
0 0 0 0 0
Suppose we have a mask kernel filter matrix

0 1 0
0 0 1
1 0 0

We multiply each pixel value of the padded matrix with the kernel and get new matrix



In the frequency domain after masking, we must convert into spatial domain frequency



youtube.com - To exit full screen, press esc

1) Multiply the input image by $(-1)^{x+y}$ to center the transform.
2) Compute $F(u,v)$, the DFT of the image from (1).
3) Multiply $F(u,v)$ by a filter function $H(u,v)$.
4) Compute the inverse DFT of the result in (3).
5) Obtain the real part of the result in (4).
6) Multiply the result by $(-1)^{x+y}$.

Pull up for precise seeking

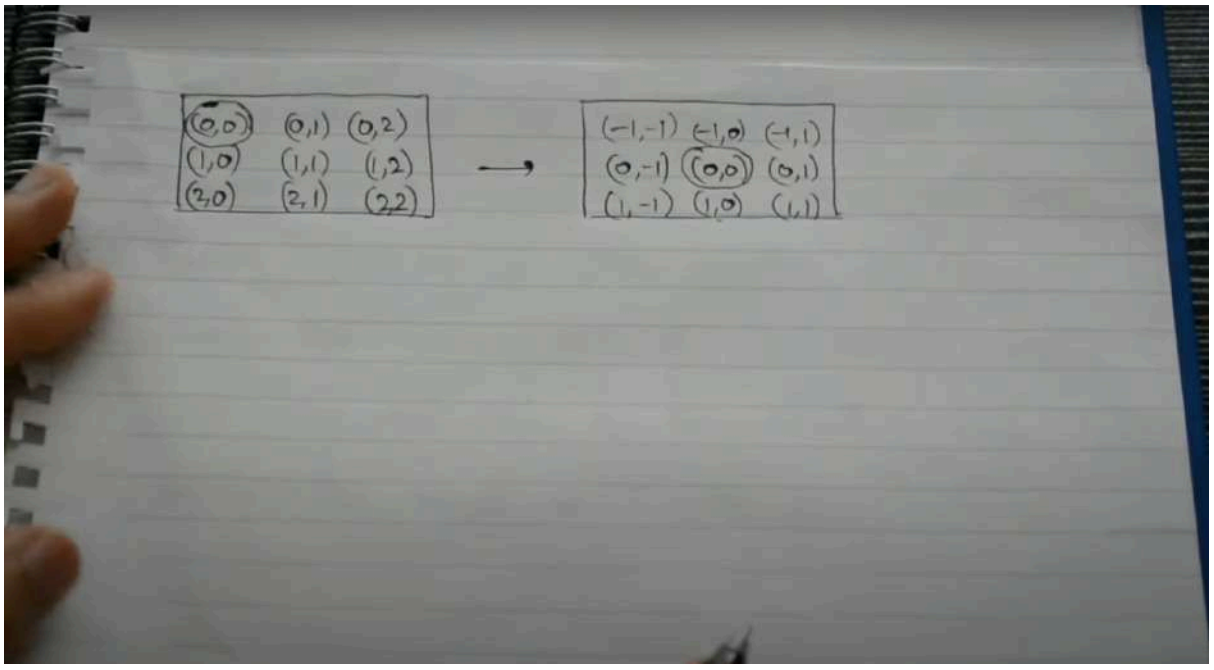
13:12

1) Multiply the input image by $(-1)^{x+y}$ to center the transform.
2) Compute $F(u,v)$, the DFT of the image from

(0,0)	(0,1)	(0,2)
(1,0)	(1,1)	(1,2)
(2,0)	(2,1)	(2,2)

→

(-1,-1)		
(0,-1)		
(1,-1)		



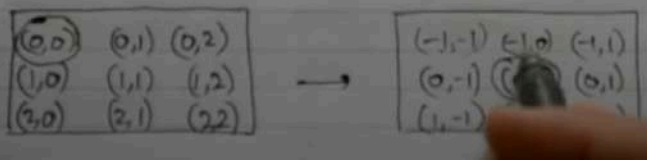
1) Multiply the input image by $(-1)^{x+y}$ to center the transform.
2) Compute $F(u,v)$, the DFT of the image from (1).
3) Multiply $F(u,v)$ by a filter function $H(u,v)$.
4) Compute the inverse DFT of the result in (3).
5) Obtain the real part of the result in (4).
6) Multiply the result in (5) by $(-1)^{x+y}$.

Smoothening frequency filters

→ These are also called low-pass filters.
→ These are used to smoothen the image as they allow only low frequency components to pass through.
→ These are used to remove noise.

Types of low-pass filters

1) Ideal low-pass filter (ILPF)
2) Butterworth low-pass filter (BLPF)
3) Gaussian low-pass filter (GLPF)



2) Butterworth low pass filter (BLPF)

$$H(u,v) = \frac{1}{1 + [D(u,v)/Do]^{2n}}$$

[where Do is the cutoff frequency and n is the order of the Butterworth filter.]

3) Gaussian low pass filter (GLPF)

$$H(u,v) = e^{-D^2(u,v)/2D_o^2}$$

[where Do is the cutoff frequency]

Sharpening frequency filters

→ These are also called high-pass filters.
→ These are used to sharpen the image as they only allow high frequency components to pass through.
→ These are used to remove background of an image.

Types of High-pass filters

1) Ideal High-pass filter (IHPF)
2) Butterworth high-pass filter (BHPF)
3) Gaussian high pass filter (GHPF)


2) Butterworth high-pass
3) Gaussian high pass

1) Ideal high-pass filter (IHPF)

$$H(u,v) = \begin{cases} 0 & \text{if } D(u,v) \le D_0 \\ 1 & \text{if } D(u,v) > D_0 \end{cases}$$

[where D_0 is the cutoff frequency]

2) Butterworth high-pass filter (BHPF)

$$H(u,v) = \frac{1}{1 + [D_0 / D(u,v)]^{2n}}$$

[where D_0 is the cutoff frequency and n is the order of the Butterworth filter.]

3) Gaussian high-pass filter (GHPF)

$$H(u,v) = 1 - e^{-D^2(u,v)/2D_0^2}$$

[where D_0 is the cutoff frequency]

$(-1)^{x+y}$

$M = 4$
$N = 4$

Note: $D(u,v) = \sqrt{\left(u - \frac{M}{2}\right)^2 + \left(v - \frac{N}{2}\right)^2} = \sqrt{(u-2)^2 + (v-2)^2}$

center $= (2,2)$.

↳ Euclidean distance

Q1. Convert the given spatial domain image using Fourier transform and perform Ideal low pass filter to smoothen the image. Choose D_0 as 0.5. Show the step by step procedure for doing the same.

1	0	1	0
1	0	1	0
1	0	1	0
1	0	1	0

Input image

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

Input image

1) Multiply the input image by $(-1)^{x+y}$ to center the transform.

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

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

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

Input image

1) Multiply the input image by $(-1)^{x+y}$ to center the transform.

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

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

$(-1)^{0+0}$ $(-1)^{0+1}$ $(-1)^{0+2}$

2) Compute the DFT of the image from (1).

$$F(u,v) = \text{kernel} \times f(x,y) \times \text{kernel}^T$$

$$= \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -j & -1 & j \\ 1 & -1 & 1 & -1 \\ 1 & j & -1 & -j \end{bmatrix} \times \begin{bmatrix} 1 & 0 & 1 & 0 \\ -1 & 0 & -1 & 0 \\ 1 & 0 & 1 & 0 \\ -1 & 0 & -1 & 0 \end{bmatrix} \times \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -j & -1 & j \\ 1 & -1 & 1 & -1 \\ 1 & j & -1 & -j \end{bmatrix}$$

$$= \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 8 & 0 & 8 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

3) Multiply $F(u,v)$ with the filter function $H(u,v)$.

For Ideal low pass filter,

$$H(u,v) = \begin{cases} 0 & \text{if } D(u,v) \le D_0 \\ 1 & \text{if } D(u,v) > D_0 \end{cases}$$

$$D(u,v) = \sqrt{\left(u - \frac{M}{2}\right)^2 + \left(v - \frac{N}{2}\right)^2}$$

$$= \sqrt{\left(u - \frac{4}{2}\right)^2 + \left(v - \frac{4}{2}\right)^2}$$

$$= \sqrt{(u-2)^2 + (v-2)^2}.$$

(Here we are calculating the distance between (u,v) from the center $(2,2)$ of the mask.)

3) Multiply $F(u,v)$ with the filter function $H(u,v)$.

For Ideal low pass filter,

$$H(u,v) = \begin{cases} 0 & \text{if } D(u,v) \le D_0 \\ 1 & \text{if } D(u,v) > D_0 \end{cases}$$

$$D(u,v) = \sqrt{\left(u - \frac{M}{2}\right)^2 + \left(v - \frac{N}{2}\right)^2}$$

$$= \sqrt{\left(u - \frac{4}{2}\right)^2 + \left(v - \frac{4}{2}\right)^2}$$

$$= \sqrt{(u-2)^2 + (v-2)^2}.$$

4×4

$M=4$, $N=4$

(0,0)	(0,1)	(0,2)	(0,3)
(1,0)	(1,1)	(1,2)	(1,3)
(2,0)	(2,1)	(2,2)	(2,3)
(3,0)	(3,1)	(3,2)	(3,3)

(Here we are calculating the distance between (u,v) from the center (2,2) of the mask.)

Distance from each value of (u,v) from (2,2):

(0,0) → $\sqrt{(0-2)^2+(0-2)^2} = \sqrt{8} = 4.24$

(0,1) → $\sqrt{(0-2)^2+(1-2)^2} = \sqrt{5} = 2.23$

(0,2) → $\sqrt{(0-2)^2+(2-2)^2} = 2$

(0,3) → $\sqrt{(0-2)^2+(3-2)^2} = \sqrt{5} = 2.23$

(1,0) → $\sqrt{(1-2)^2+(0-2)^2} = \sqrt{5} = 2.23$

(1,1) → $\sqrt{(1-2)^2+(1-2)^2} = \sqrt{2} = 1.41$

(1,2) → $\sqrt{(1-2)^2+(2-2)^2} = 1$

(1,3) → $\sqrt{(1-2)^2+(3-2)^2} = \sqrt{2} = 1.41$

(2,0) → $\sqrt{(2-2)^2+(0-2)^2} = 2$

(2,1) → $\sqrt{(2-2)^2+(1-2)^2} = 1$

(2,2) → $\sqrt{(2-2)^2+(2-2)^2} = 0$

(2,3) → $\sqrt{(2-2)^2+(3-2)^2} = 1$

(3,0) → $\sqrt{(3-2)^2+(0-2)^2} = \sqrt{5} = 2.23$

(3,1) → $\sqrt{(3-2)^2+(1-2)^2} = \sqrt{2} = 1.41$

(3,2) → $\sqrt{(3-2)^2+(2-2)^2} = 1$

(3,3) → $\sqrt{(3-2)^2+(3-2)^2} = \sqrt{2} = 1.41$

$(2,2) \rightarrow \sqrt{(2-2)^2 + (2-2)^2} = 0$

$(2,3) \rightarrow \sqrt{(2-2)^2 + (3-2)^2} = 1$

$(3,0) \rightarrow \sqrt{(3-2)^2 + (0-2)^2} = \sqrt{5} = 2.23$

$(3,1) \rightarrow \sqrt{(3-2)^2 + (1-2)^2} = \sqrt{2} = 1.41$

$(3,2) \rightarrow \sqrt{(3-2)^2 + (2-2)^2} = 1$

$(3,3) \rightarrow \sqrt{(3-2)^2 + (3-2)^2} = \sqrt{2} = 1.41.$

$$D(u,v) = \begin{bmatrix} 4.24 & 2.23 & 2 & 2.23 \\ 2.23 & 1.41 & 1 & 1.41 \\ 2 & 1 & 0 & 1 \\ 2.23 & 1.41 & 1 & 1.41 \end{bmatrix}.$$

$D_0 = 0.5$ (Given)

$$\therefore H(u,v) = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix}$$

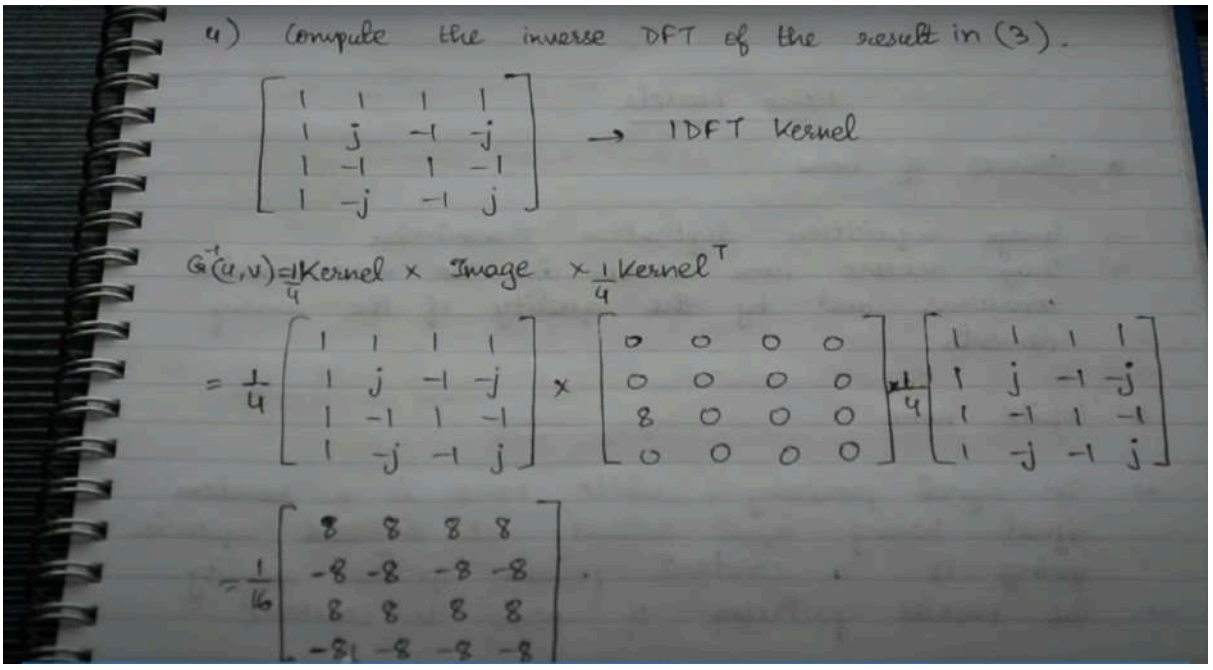
$$= \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 8 & 0 & 8 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 8 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}.$$

4) Compute the inverse DFT of the result in (3).

$$\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & j & -1 & -j \\ 1 & -1 & 1 & -1 \\ 1 & -j & -1 & j \end{bmatrix} \rightarrow \text{IDFT Kernel}$$

$G^{-1}(u,v) = \frac{1}{4}\text{Kernel} \times \text{Image} \times \frac{1}{4}\text{Kernel}^T$



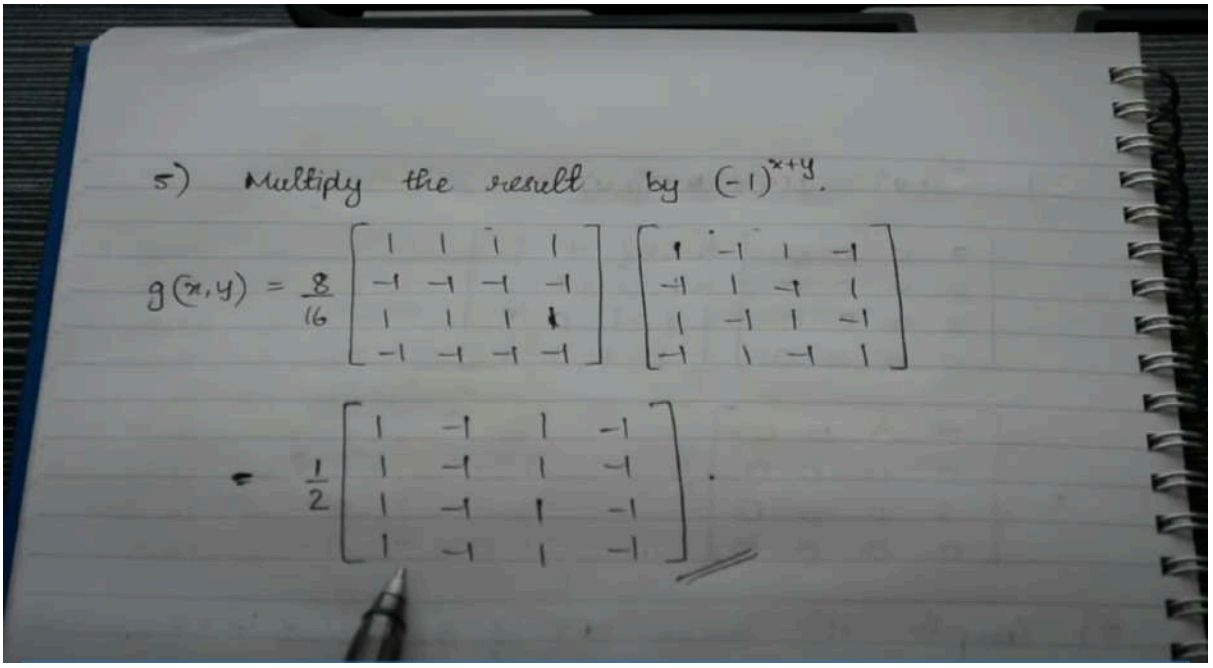
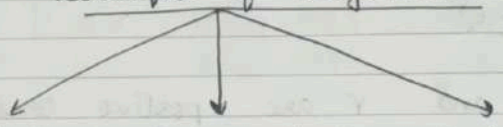


Image Enhancement Using Histogram Equalization

Image Enhancement

The process of manipulating the image so that the result is more suitable than the original for specific applications.

Techniques of Image Enhancement



Spatial domain Involves direct manipulation of pixels.

Frequency domain Involves modifying the Fourier transform of an image.

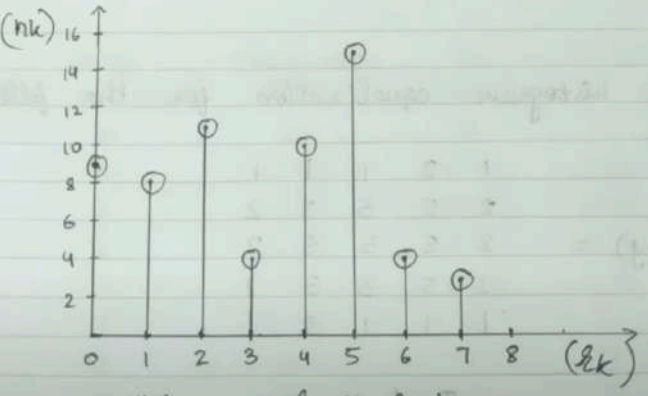
Combination Involves a combination of spatial and frequency domains.

Spatial Domain

1) Histogram Equalization

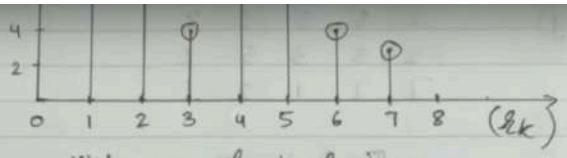
Q.1. Perform the histogram equalization for an 8 × 8 image shown below.

Gray levels	0	1	2	3	4	5	6	7
No. of pixels	9	8	11	4	10	15	4	3



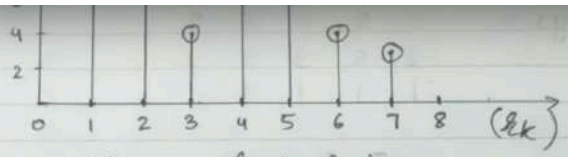
Histogram of input image

Gray	No of	$p(r_k) = n_k/n$	s_k	$s_k \times 7$	Histogram



Histogram of input image

Gray levels (r_k)	No of pixels (n_k)	$p(r_k) = n_k/n$ (PDF)	s_k (CDF)	$s_k \times 7$	Histogram Equalization level
0	9	0.141	0.141	0.987	1
1	8	0.125	0.266	1.862	2
2	11	0.172	0.438	3.066	3
3	4	0.0625	0.5005	3.5035	4
4	10	0.156	0.6565	4.5955	5
5	15	0.234	0.8905	6.2335	6
6	4	0.0625	0.953	6.671	7
7	3	0.047	1	7	7
	$n = 64$				

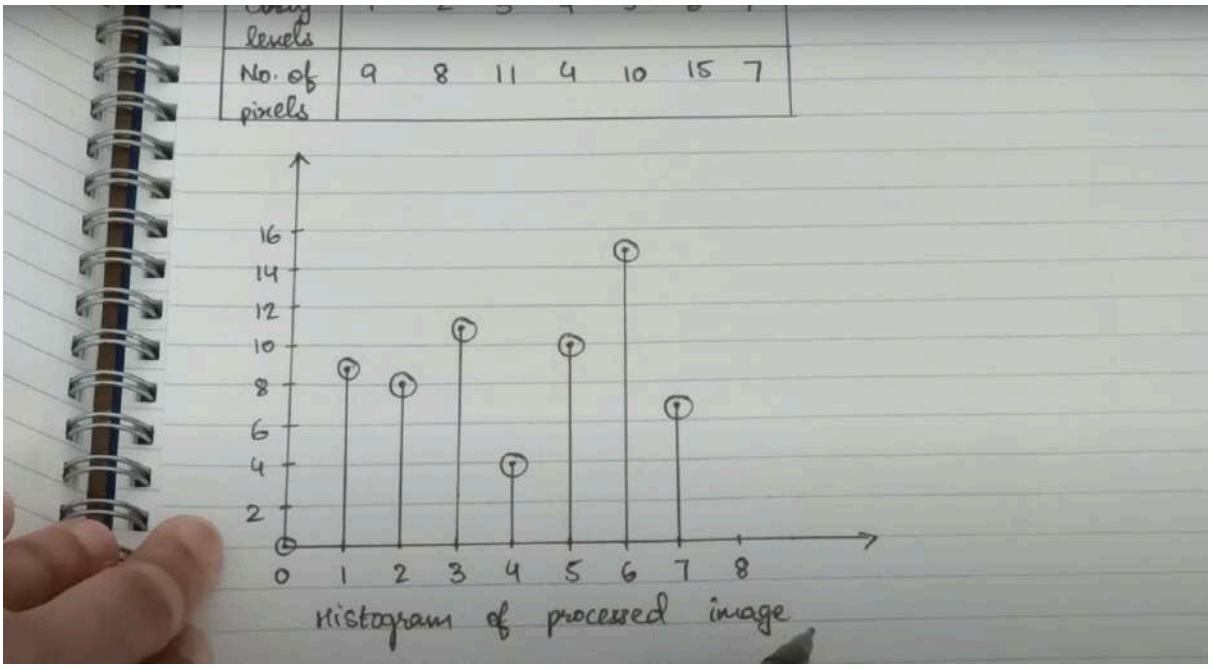


Histogram of input image [0-7]

Gray levels (r_k)	No. of pixels (n_k)	$p(r_k) = n_k/n$ (PDF)	s_k (CDF)	$s_k \times 7$	Histogram Equalization level
0	9	0.141	0.141	0.987	→ 1
1	8	0.125	0.266	1.862	→ 2
2	11	0.172	0.438	3.066	→ 3
3	4	0.0625	0.5005	3.5035	→ 4
4	10	0.156	0.6565	4.5955	5
5	15	0.234	0.8905	6.2335	6
6	4	0.0625	0.953	6.671	7
7	3	0.047	1	7	7
	n = 64				

Gray levels (r_k)	No. of pixels (n_k)	$p(r_k) = n_k/n$ (PDF)	s_k (CDF)	$s_k \times 7$	Histogram Equalization level
0	9	0.141	0.141	0.987	→ 1
1	8	0.125	0.266	1.862	→ 2
2	11	0.172	0.438	3.066	→ 3
3	4	0.0625	0.5005	3.5035	→ 4
4	10	0.156	0.6565	4.5955	5
5	15	0.234	0.8905	6.2335	6
6	4	0.0625	0.953	6.671	7
7	3	0.047	1	7	7
	n = 64				

Gray levels	1	2	3	4	5	6	7
No. of pixels	9	8	11	4	10	15	7


levels
No. of pixels
9 8 11 4 10 15 7
16
14
12
10
8
6
4
2
0 1 2 3 4 5 6 7 8
Histogram of processed image

Q.2. Perform histogram equalization for the following image.

$$f(x,y) = \begin{bmatrix} 1 & 2 & 1 & 1 & 1 \\ 2 & 5 & 3 & 5 & 2 \\ 2 & 5 & 5 & 5 & 2 \\ 2 & 5 & 3 & 5 & 2 \\ 1 & 1 & 1 & 2 & 1 \end{bmatrix}$$

Max. value = 5.

Input image

Gray levels (r_k)	0	1	2	3	4	5	6	7
No. of pixels (n_k)	0	8	8	2	0	7	0	0

Q.2. Perform histogram equalization for the following image.

$$f(x,y) = \begin{bmatrix} 1 & 2 & 1 & 1 & 1 \\ 2 & 5 & 3 & 5 & 2 \\ 2 & 5 & 5 & 5 & 2 \\ 2 & 5 & 3 & 5 & 2 \\ 1 & 1 & 1 & 2 & 1 \end{bmatrix}$$

Max. value = 5.

Input image

$2^0 = 1$

$2^1 = 2$

$2^2 = 4$

$2^3 = 8$

Gray levels (r_k)	0	1	2	3	4	5	6
No. of pixels (n_k)	0	8	8	2	0	7	0

$[0, L-1]$

Q.2. Perform histogram equalization for the following image.

$$f(x,y) = \begin{bmatrix} 1 & 2 & 1 & 1 & 1 \\ 2 & 5 & 3 & 5 & 2 \\ 2 & 5 & 5 & 5 & 2 \\ 2 & 5 & 3 & 5 & 2 \\ 1 & 1 & 1 & 2 & 1 \end{bmatrix}$$

Max. value = 5.

$2^0 = 1$

$2^1 = 2$

$2^2 = 4$

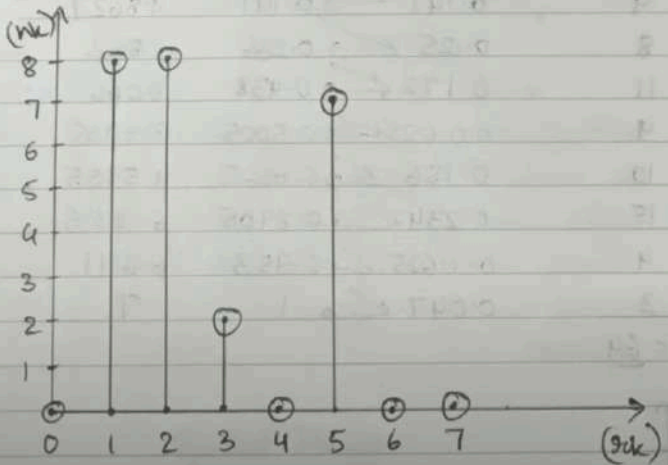
$2^3 = 8$ $L = 8$

$L-1 = 7$

Input image

Gray levels (r_k)	0	1	2	3	4	5	6	7
No. of pixels (n_k)	0	8	8	2	0	7	0	0

Gray levels (r_k)	0	1	2	3	4	5	6	7
No. of pixels (n_k)	0	8	8	2	0	7	0	0



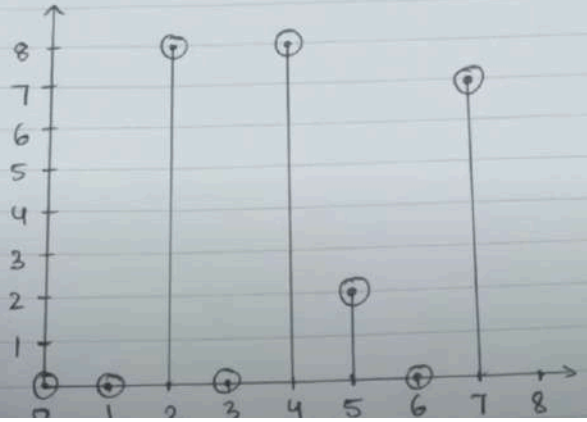
Histogram of input image

Gray levels (r_k)	No of pixels (n_k)	$P(r_k) = n_k/n$ (PDF)	S_k (CDF)	$S_k \times 7$	Histogram Equalization level
0	0	0	0	0	0
1	8	0.32	0.32	2.24	2
2	8	0.32	0.64	4.48	4
3	2	0.08	0.72	5.04	5
4	0	0	0.72	5.04	5
5	7	0.28	1	7	7
6	0	0	1	7	7
(7)	0	0	1	7	7

$n = 25$

Gray levels (r_k)	No of pixels (n_k)	$P(r_k) = n_k/n$ (PDF)	S_k (CDF)	$S_k \times 7$	Histogram Equalization level
0	0	0	0	0	0
1	8	0.32	0.32	2.24	2
2	8	0.32	0.64	4.48	4
3	2	0.08	0.72	5.04	5
4	0	0	0.72	5.04	5
5	7	0.28	1	7	7
6	0	0	1	7	7
(7)	0	0	1	7	7

Gray levels	0	2	4	5	7
No. of pixels	0	8	8	2	7



0	0	0 →	0	0 → 0
1	8	0.32 ⟷	0.32	2.24 → 2
2	8	0.32 ⟷	0.64	4.48 → 4
3	2]	0.08 ⟷	0.72	5.04 → 5
4	0]	0	0.72	5.04 → 5
5	7]	0.28	1	7 → 7
6	0]	0	1	7 → 7
(7)	0]	0	1	7 → 7

1	2	1	1	1
2	5	3	5	2
2	5	5	5	2
2	5	3	5	2
1	1	1	2	1

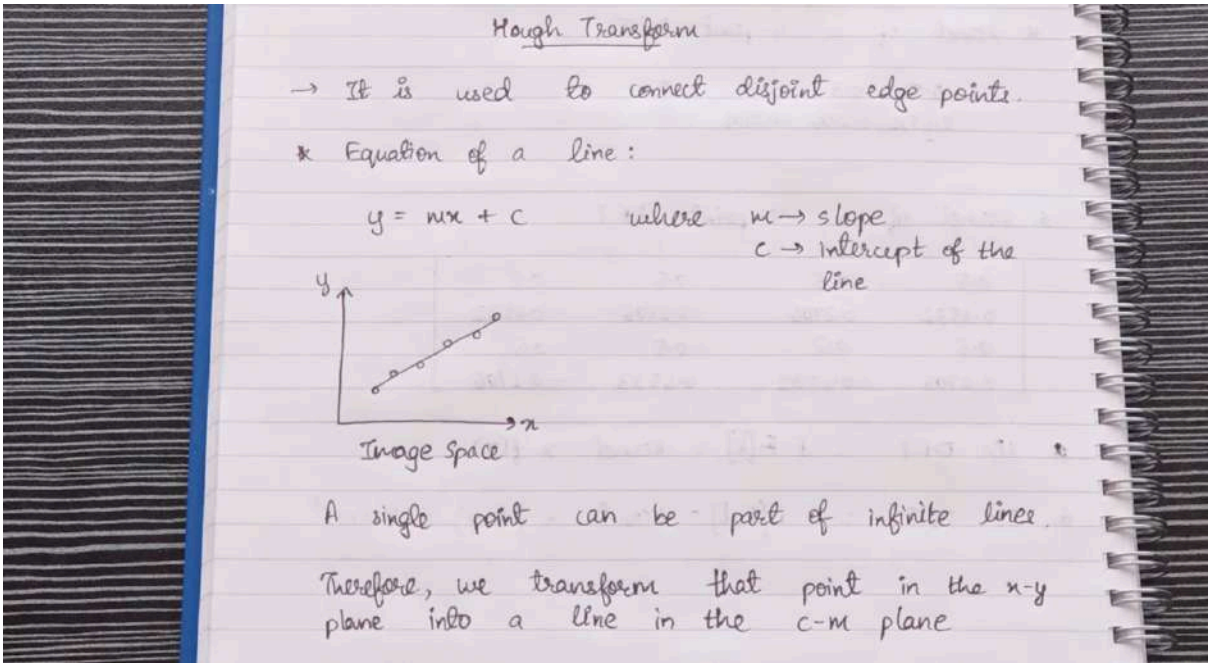
→

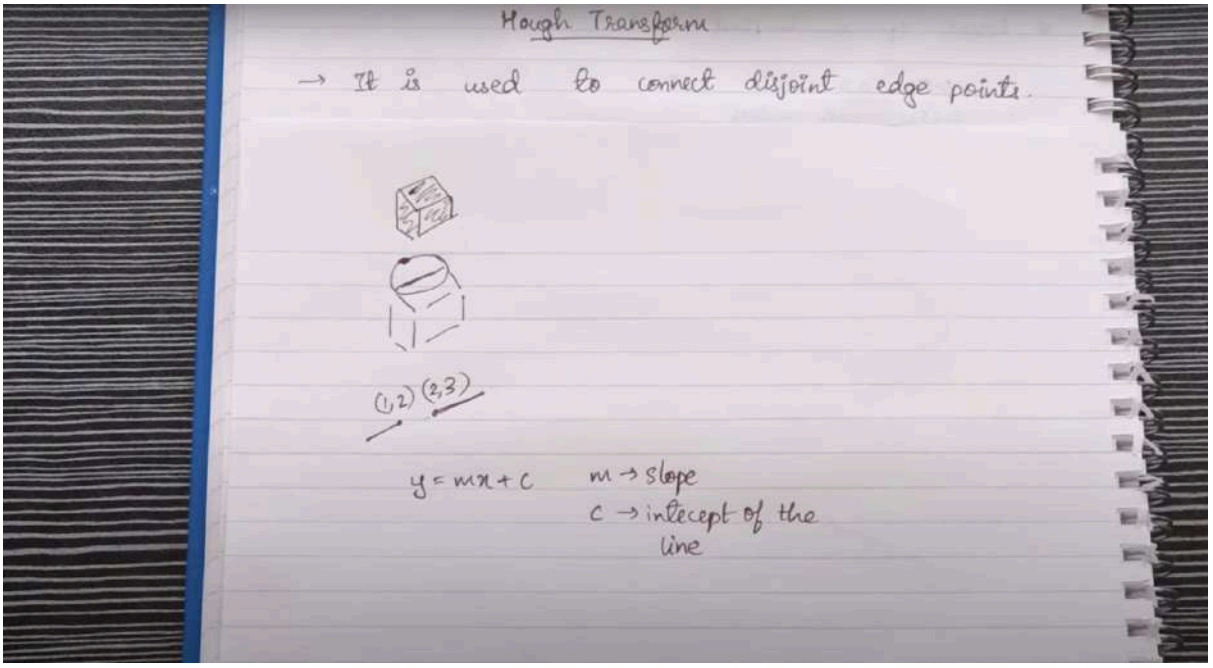
2	4	2	2	2
4	7	5	7	4
4	7	7	7	4
4	7	5	7	4
2	2	2	4	2

Input Image

Output Image

Hough Transformation - Line Detection and Edge Detection

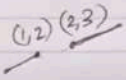




It is also used to join the line two lines as an edge-detection
How many straight lines pass through the center dot point?
Line that interact with each other that makes them edge?

Hough Transform

→ It is used to connect disjoint edge points.



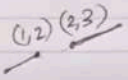
$y = mx + c$

$m \rightarrow$ slope

$c \rightarrow$ intercept of the line

Hough Transform

→ It is used to connect disjoint edge points.

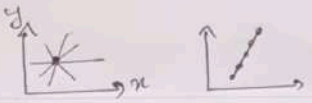


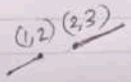
$y = mx + c$ $m \rightarrow$ slope

$c \rightarrow$ intercept of the line

Hough Transform

→ It is used to connect disjoint edge points.



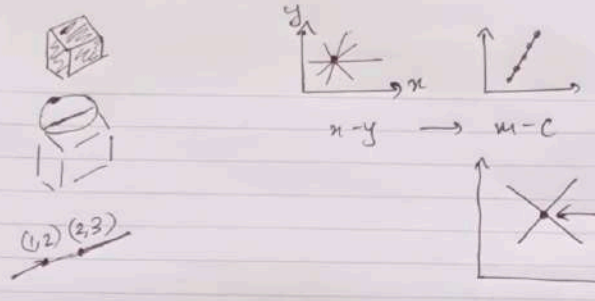


$y = mx + c$ $m \rightarrow$ slope

$c \rightarrow$ intercept of the line

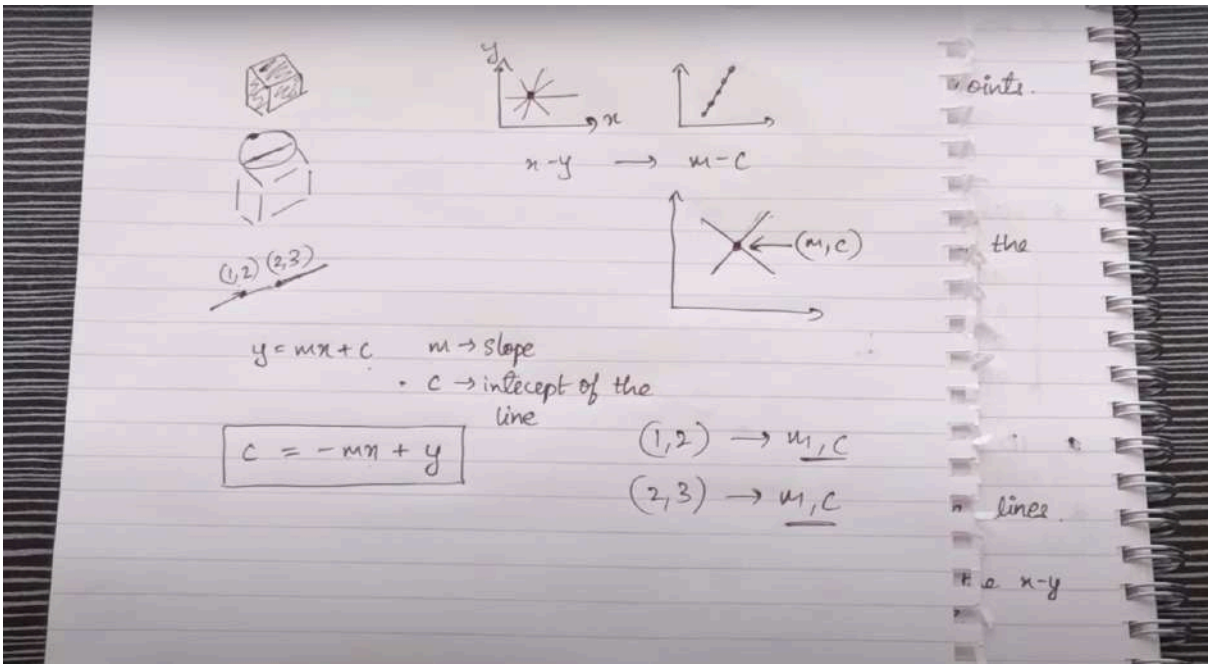
Hough Transform

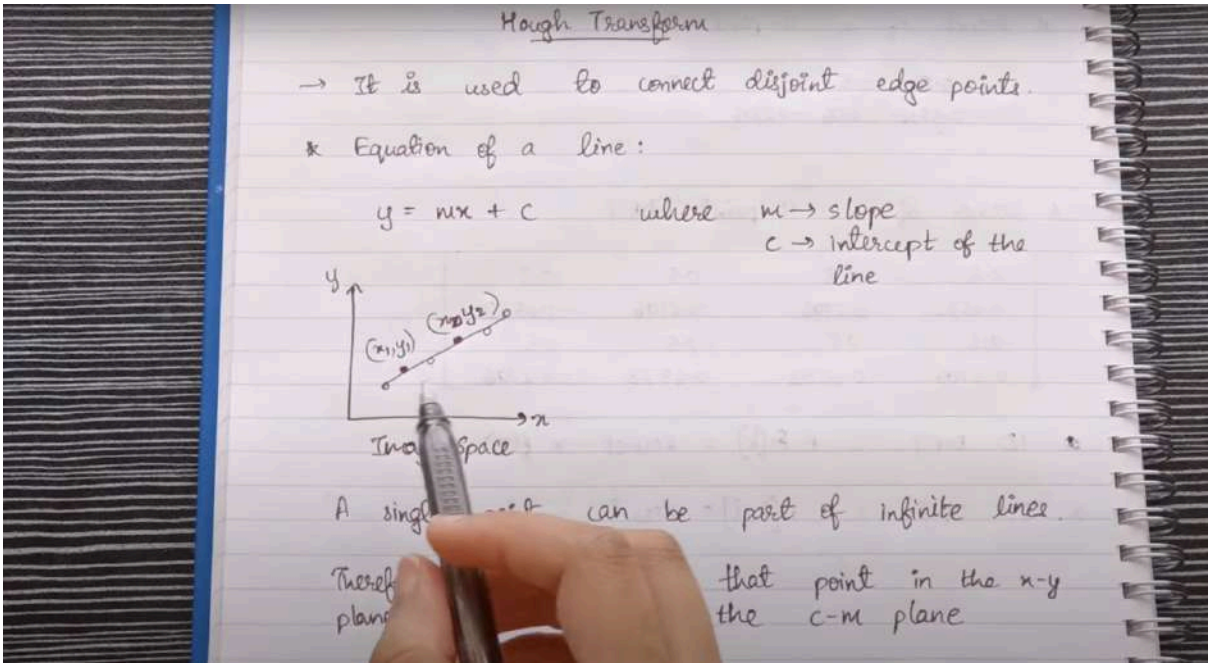
→ It is used to connect disjoint edge points.

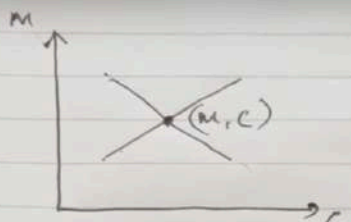


$y = mx + c$ $m \rightarrow$ slope

- $c \rightarrow$ intercept of the line


y
x
x-y → m-c
(m,c)
(1,2) (2,3)
y = mx + c
m → slope
c → intercept of the line
c = -mx + y
(1,2) → m, c
(2,3) → m, c
oints.
the
lines.
i.e. x-y


Hough Transform
→ It is used to connect disjoint edge points.
* Equation of a line:
y = mx + c where m → slope
c → intercept of the line
y
x
(x1,y1)
(x2,y2)
Image Space
A single point can be part of infinite lines.
Therefore that point in the x-y plane the c-m plane



Parameter space
(or)
Hough space

$(x_1, y_1) \longrightarrow (m_1, c_1)$

$(x_2, y_2) \longrightarrow (m_2, c_2)$

If A and B are two points connected by a line in the spatial domain, they will be intersecting lines in the Hough space.

(i) For $(x, y) = (1, 2)$, $c = -m + 2$

if $c = 0$, $0 = -m + 2$

$\boxed{m = 2}$

if $m = 0$, $\boxed{c = 2}$

Thus, $(m, c) = (2, 2)$.

(ii) For $(x, y) = (2, 3)$, $c = -2m + 3$.

if $c = 0$, $0 = -2m + 3$

$2m = 3$

$m = 3/2 = 1.5$. $\boxed{m = 1.5}$

if $m = 0$, $\boxed{c = 3.}$

Thus, $(m, c) = (1.5, 3)$.

$m = 3/2 = 1.5$. $\boxed{m = 1.5}$

if $m = 0$, $\boxed{c = 3}$.

Thus, $(m, c) = (1.5, 3)$.

(iii) For $(x, y) = (3, 4)$, $c = -3m + 4$.

if $c = 0$, $0 = -3m + 4$

$3m = 4$

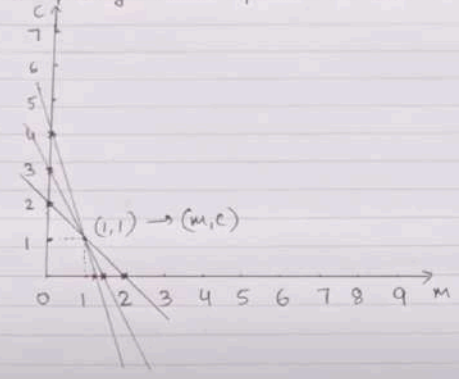
$m = 4/3 = 1.33$. $\boxed{m = 1.33}$

if $m = 0$, $\boxed{c = 4}$.

Thus $(m, c) = (1.33, 4)$.

$(m, c) = (2, 2), (1.5, 3), (1.33, 4)$.

On plotting these points in the m-c plane:

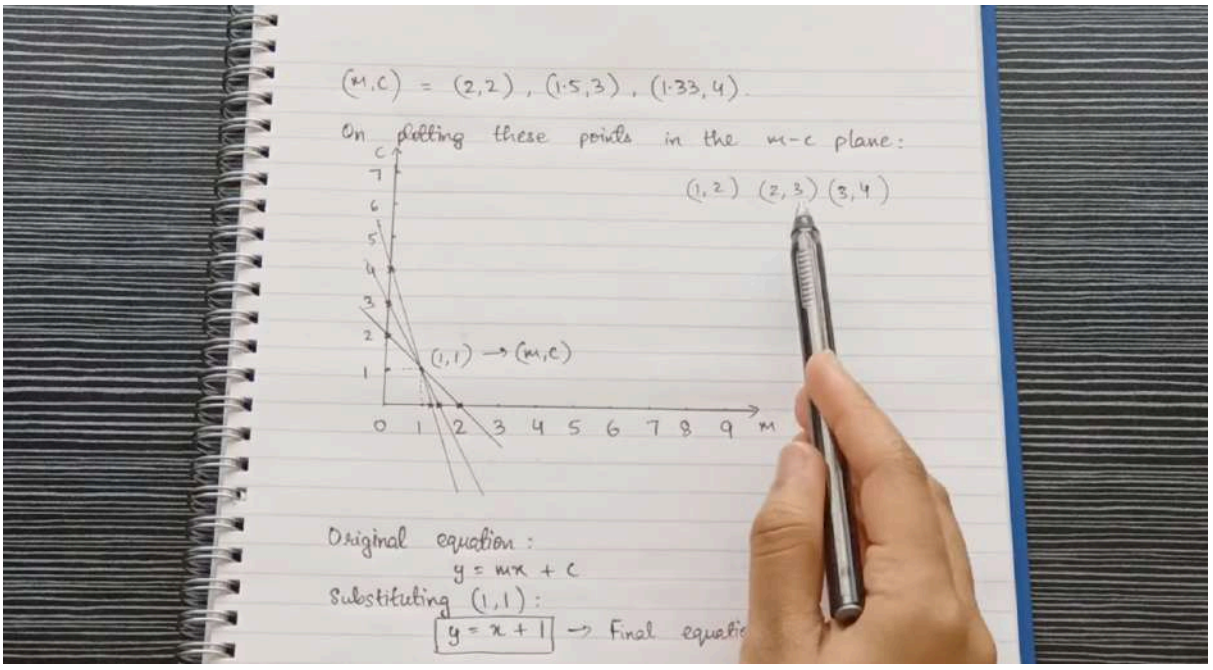


Original equation:

$y = mx + c$

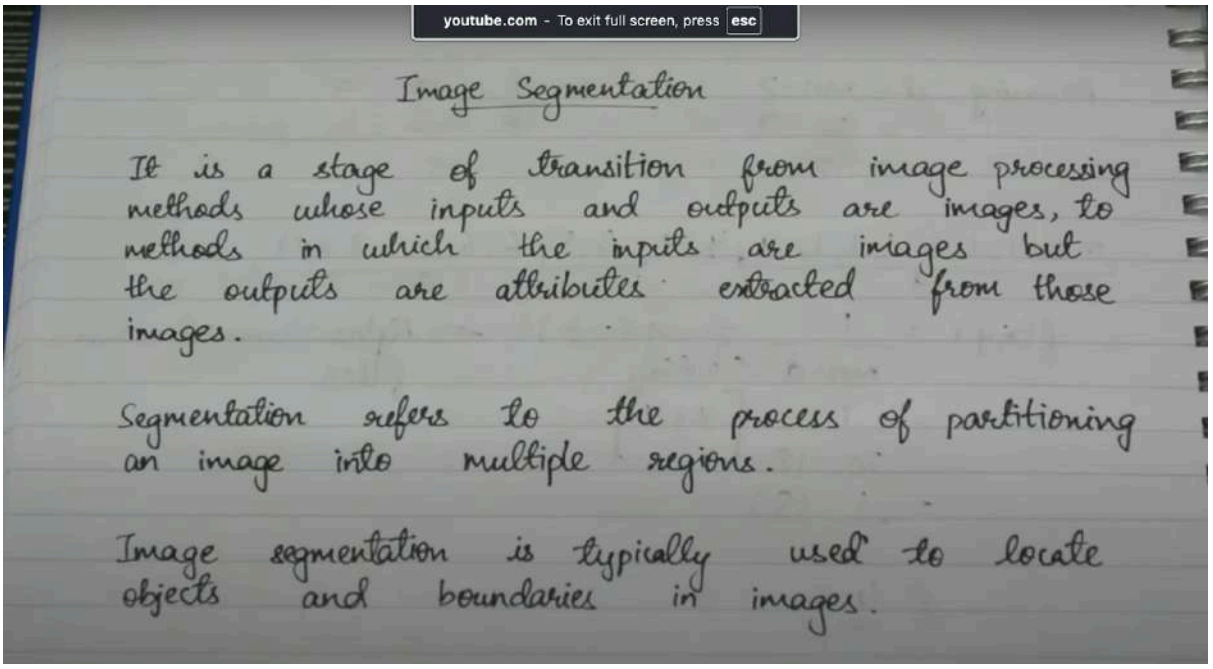
Substituting (1,1):

$\boxed{y = x + 1}$

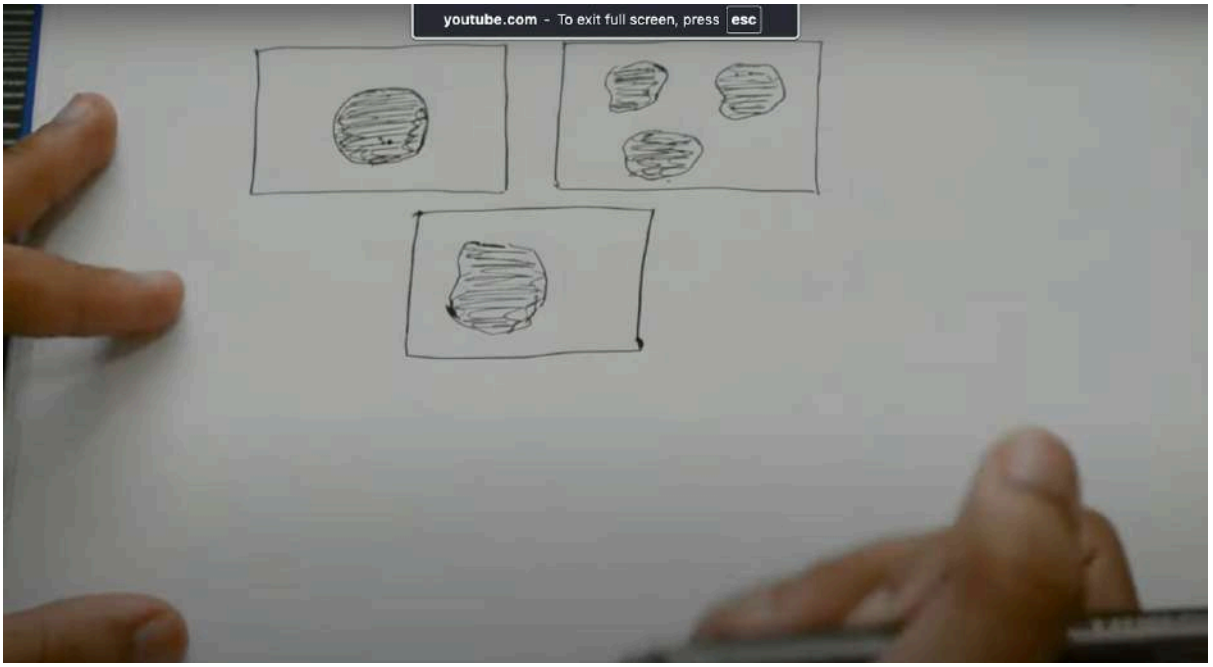


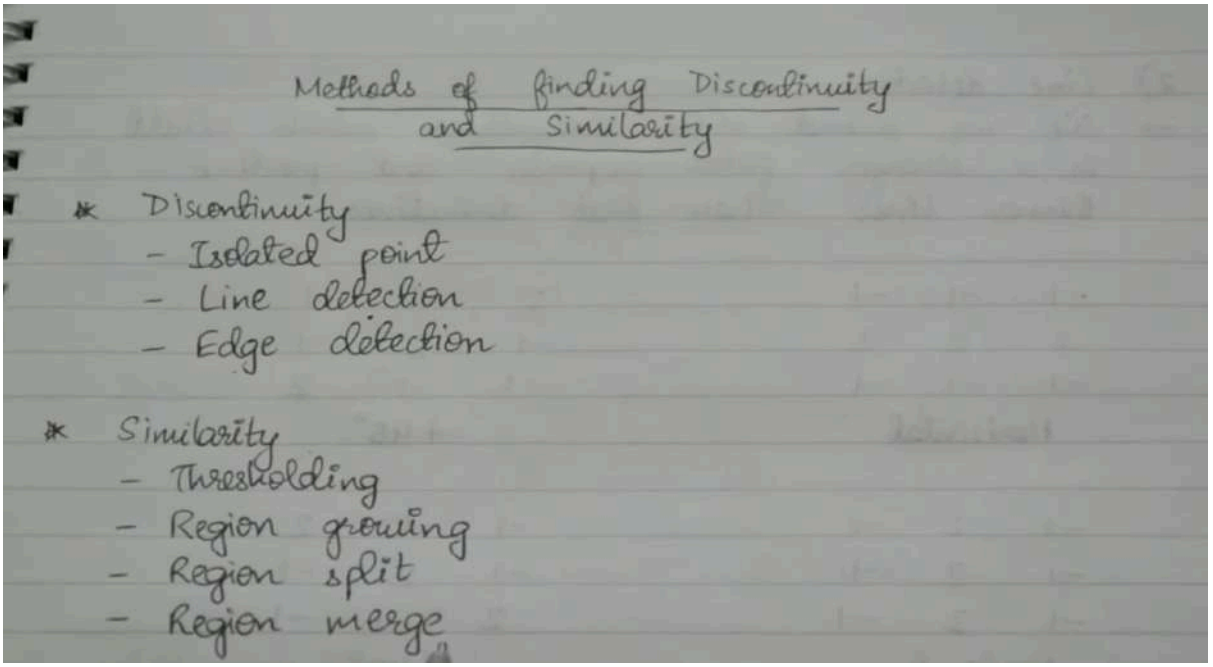
Y = 1 + 1
Y = 2+1 proved

Image Segmentation Methods



Edge, boundary, line detection





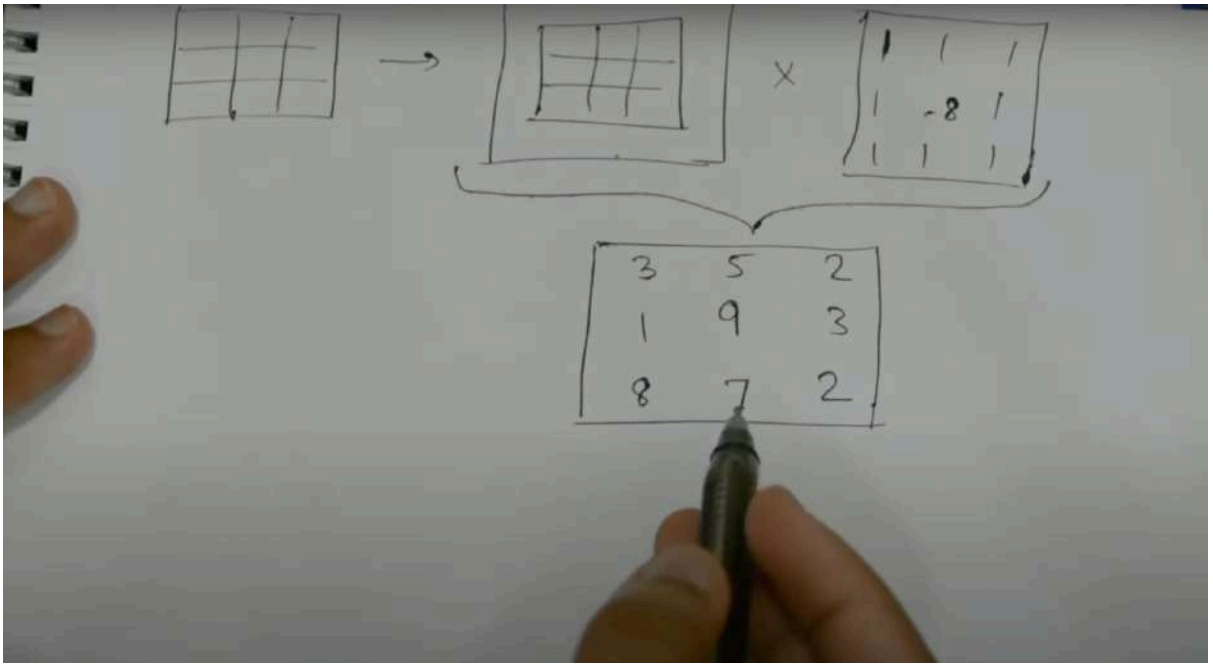
→ A point has been detected at a location (x, y) on which the kernel is centered if the absolute value of the response of the filter at that point exceeds a specified threshold.

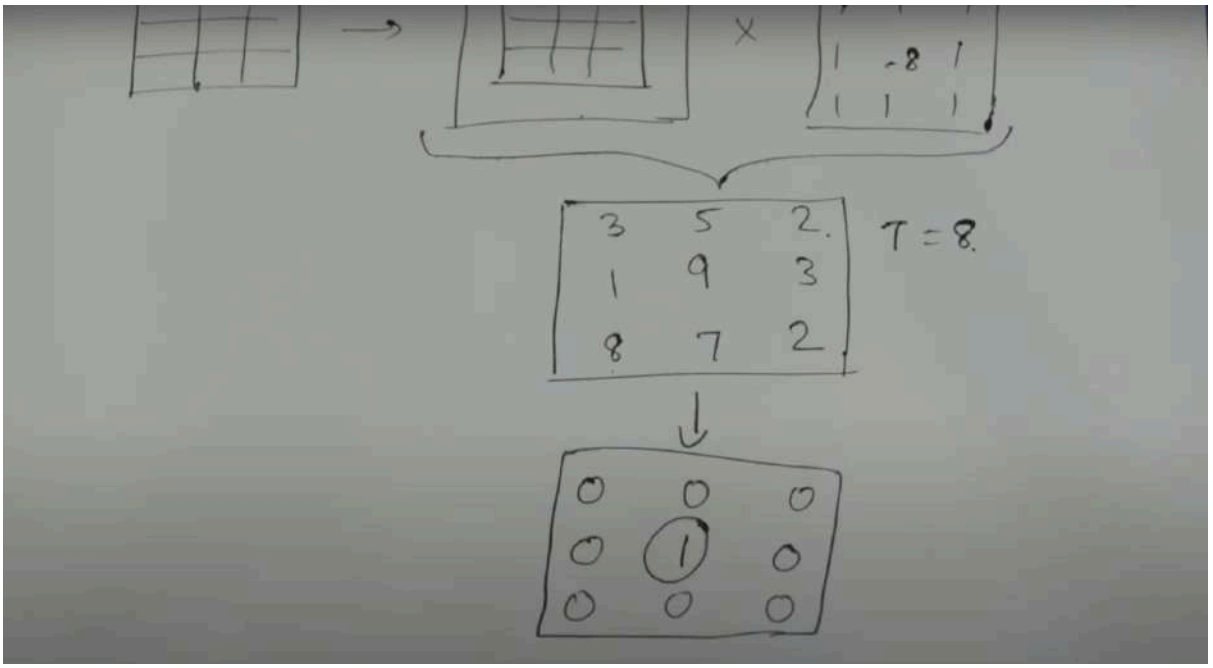
→ Such points are labeled 1 and all others are labeled 0 in the output image, thus producing a binary image.

1	1	1
1	-8	1
1	1	1

Kernel

$$g(x,y) = \begin{cases} 1 & \text{if } |Z(x,y)| > T \\ 0 & \text{otherwise} \end{cases}$$





→ A point has been detected at a location (x, y) on which the kernel is centered if the absolute value of the response of the filter at that point exceeds a specified threshold.

→ Such points are labeled 1 and all others are labeled 0 in the output image, thus producing a binary image.

1	1	1
1	-8	1
1	1	1

Kernel

$$g(x,y) = \begin{cases} 1 & \text{if } |Z(x,y)| > T \\ 0 & \text{otherwise} \end{cases}$$

2) Line detection

→ We use second order derivatives which result in a stronger filter response and produce thinner lines than first derivatives.

-1	-1	-1
2	2	2
-1	-1	-1

Horizontal

2	-1	-1
-1	2	-1
-1	-1	2

+45°

-1	2	-1
-1	2	-1
-1	2	-1

Vertical

-1	-1	2
-1	2	-1
2	-1	-1

-45°

3) Edge detection

→ It is an approach used frequently for segmenting images based on abrupt (local) changes in intensity.

→ First order derivatives such as Roberts-cross, Prewitt and Sobel operators are preferred for thicker lines.

→ Second order derivatives (laplacian) are used for detecting thinner lines

→ Kirsch compass kernels are used for finding the maximum edge strength in a few predetermined directions.

Edge Detection Kernels

1) Roberts cross-gradient operators

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

2) Prewitt operators

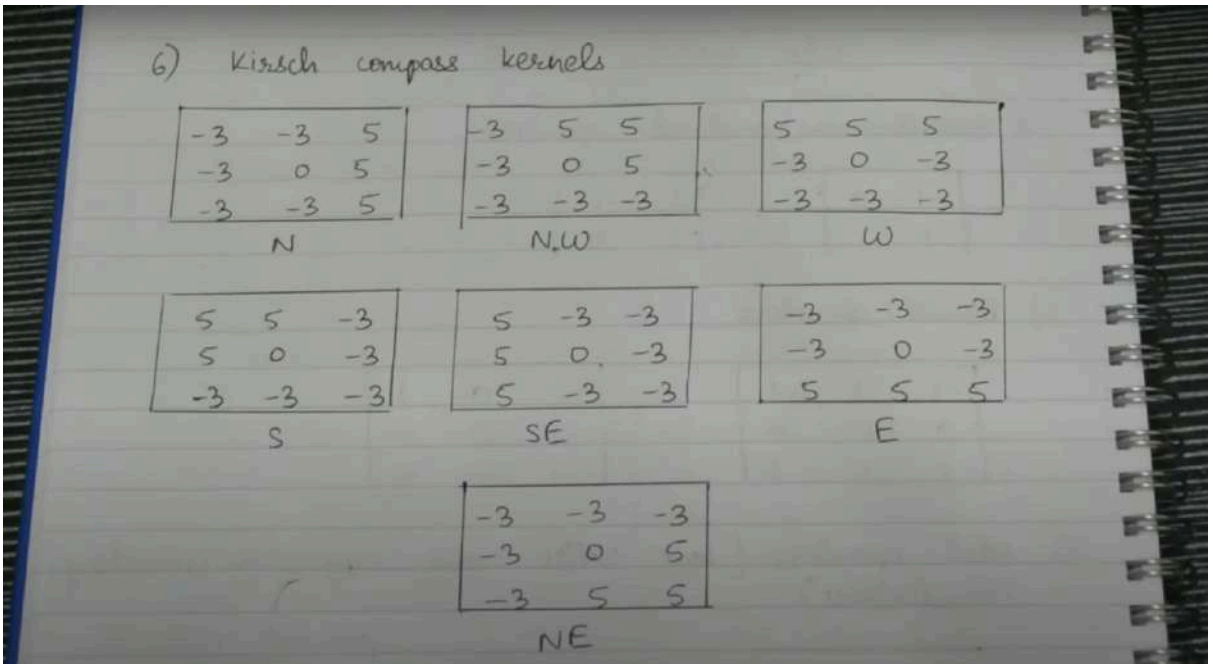
$$\begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix} \qquad \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}$$

4) Prewitt masks for diagonal edges

$$\begin{bmatrix} 0 & 1 & 1 \\ -1 & 0 & 1 \\ -1 & -1 & 0 \end{bmatrix} \quad +45^\circ \qquad \begin{bmatrix} -1 & -1 & 0 \\ -1 & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix} \quad -45^\circ$$

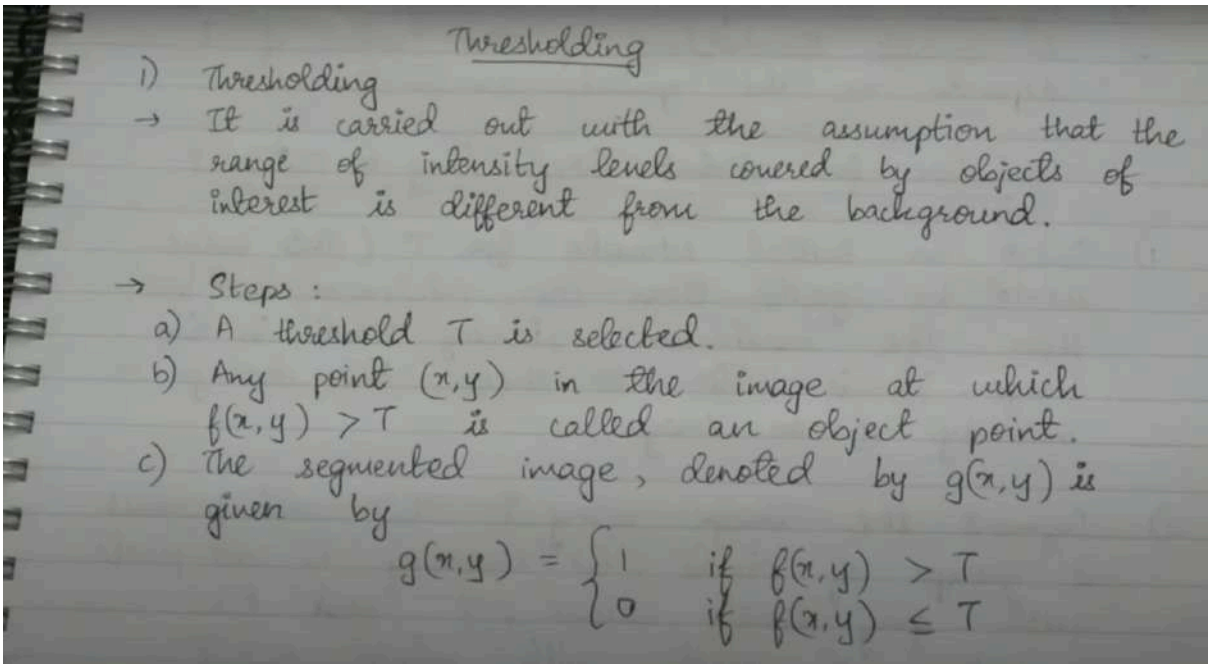
5) Sobel masks for diagonal edges

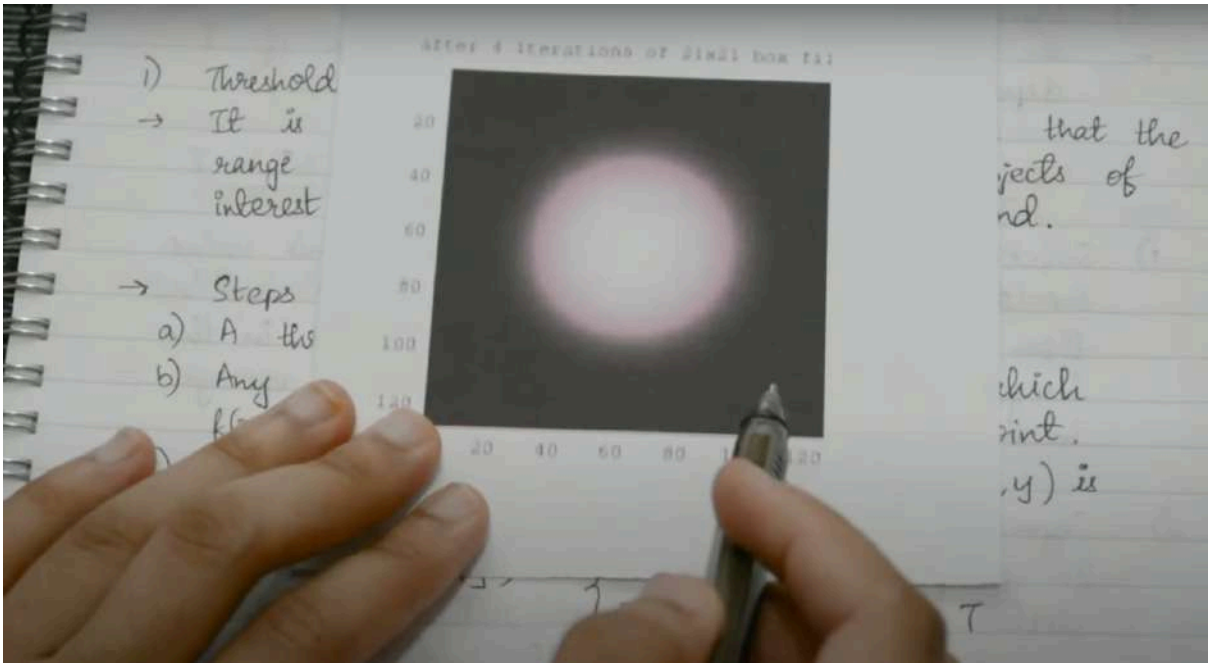
$$\begin{bmatrix} 0 & 1 & 2 \\ -1 & 0 & 1 \\ -2 & -1 & 0 \end{bmatrix} \quad +45^\circ \qquad \begin{bmatrix} -2 & -1 & 0 \\ -1 & 0 & 1 \\ 0 & 1 & 2 \end{bmatrix} \quad -45^\circ$$



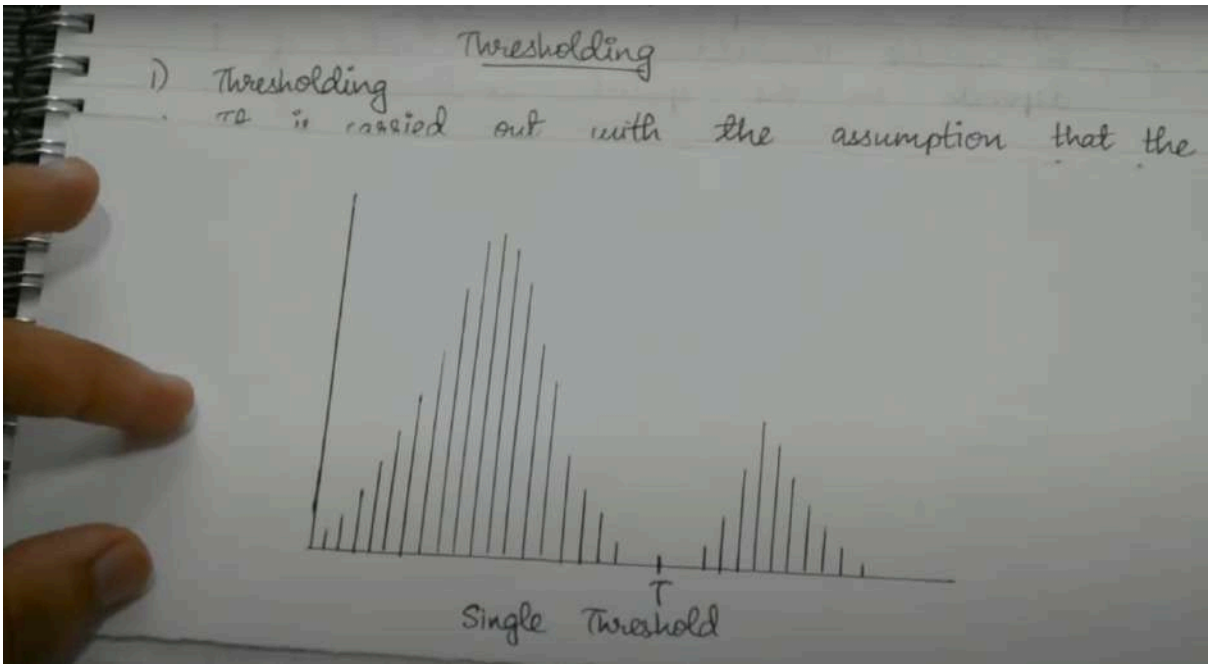
Thresholding in Image Processing

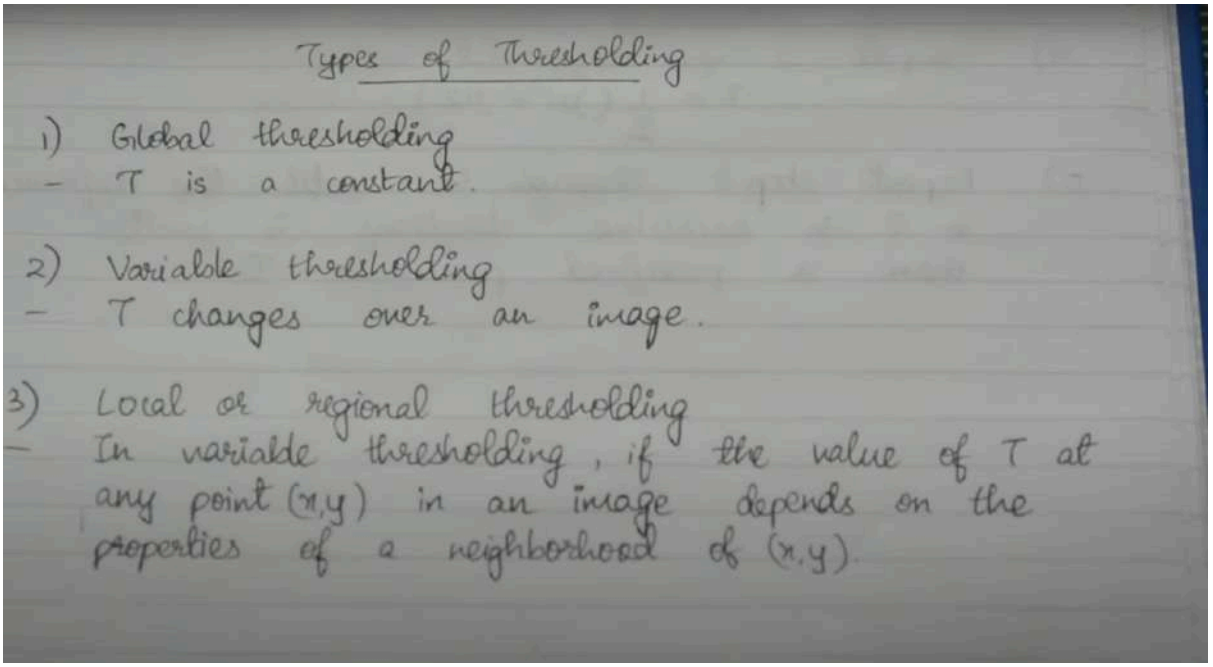
Either a black background and white image or an image black and background white





Two dominant models so the single threshold





See from the backward

"""

Diagram Explanation:

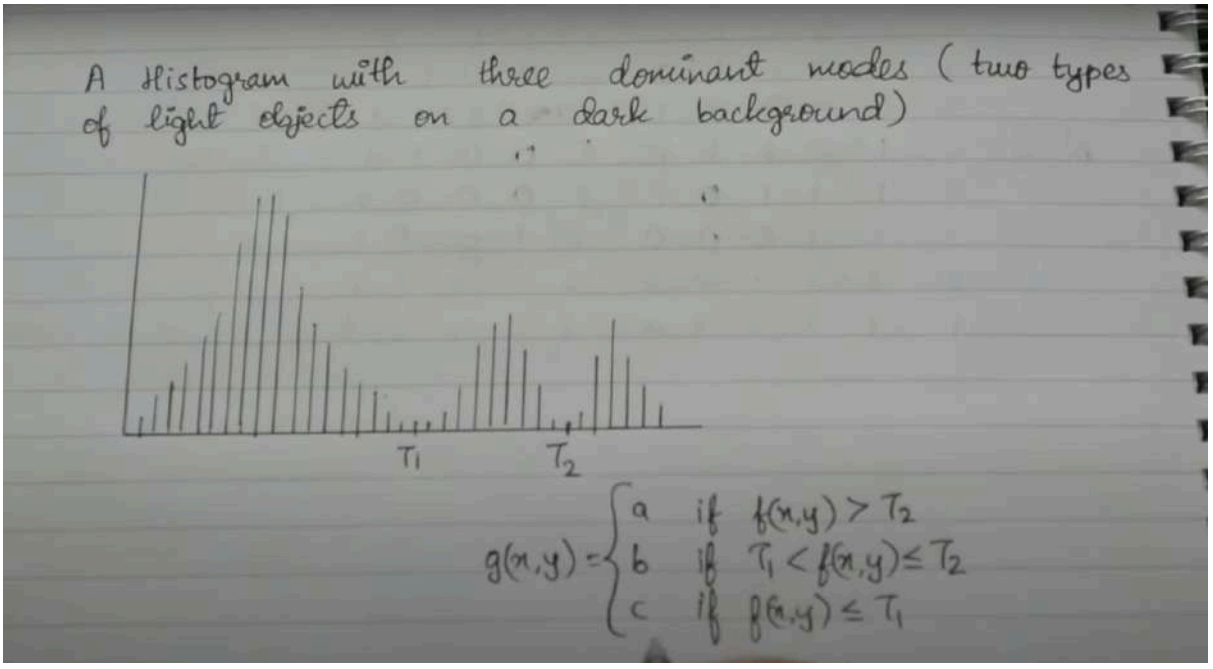
1. Histogram with Three Modes:
   - Represents intensity levels of an image with two types of light objects on a dark background.
   - Left peak: **Dark background.**
   - Middle peak: **First type of light object.**

- Right peak: **Second, brighter type of light object.**

2. Thresholds T1 and T2:
   - **T1**: **Separates background from the first light object.**
   - **T2**: **Separates the first light object from the second, brighter object**.

3. Piecewise Function g(x, y):
   - g(x, y) = a if f(x, y) > T2 (brightest object).
   - g(x, y) = b if T1 < f(x, y) <= T2 (second brightest object).
   - g(x, y) = c if f(x, y) <= T1 (background).

"""

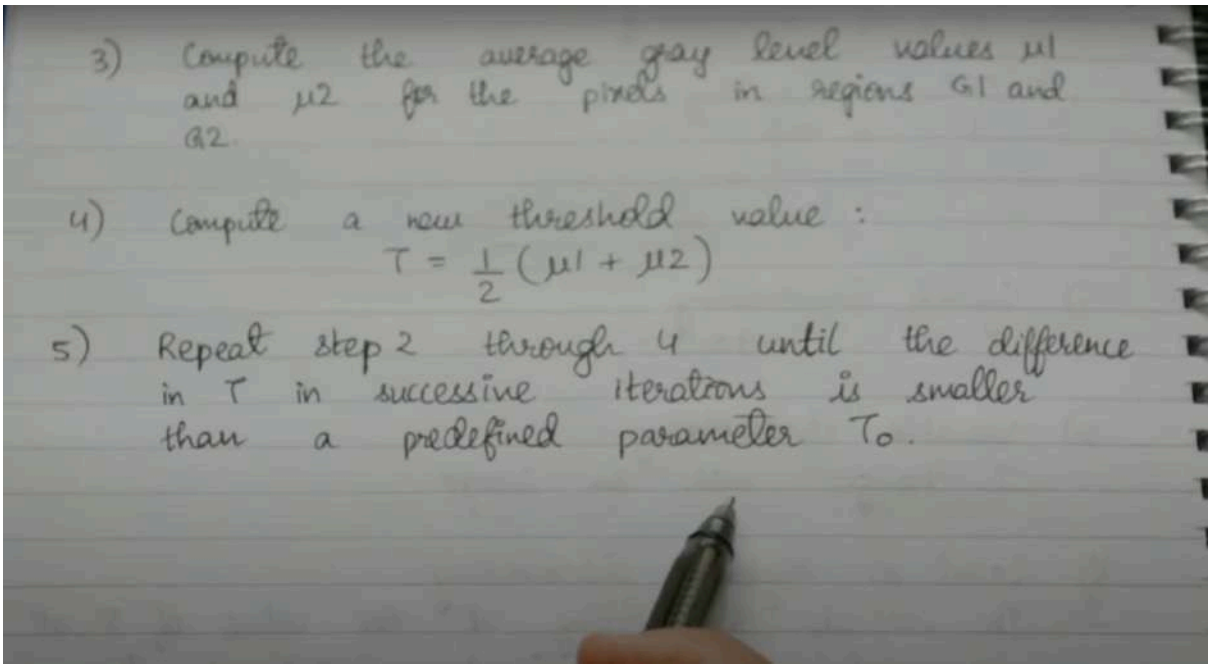


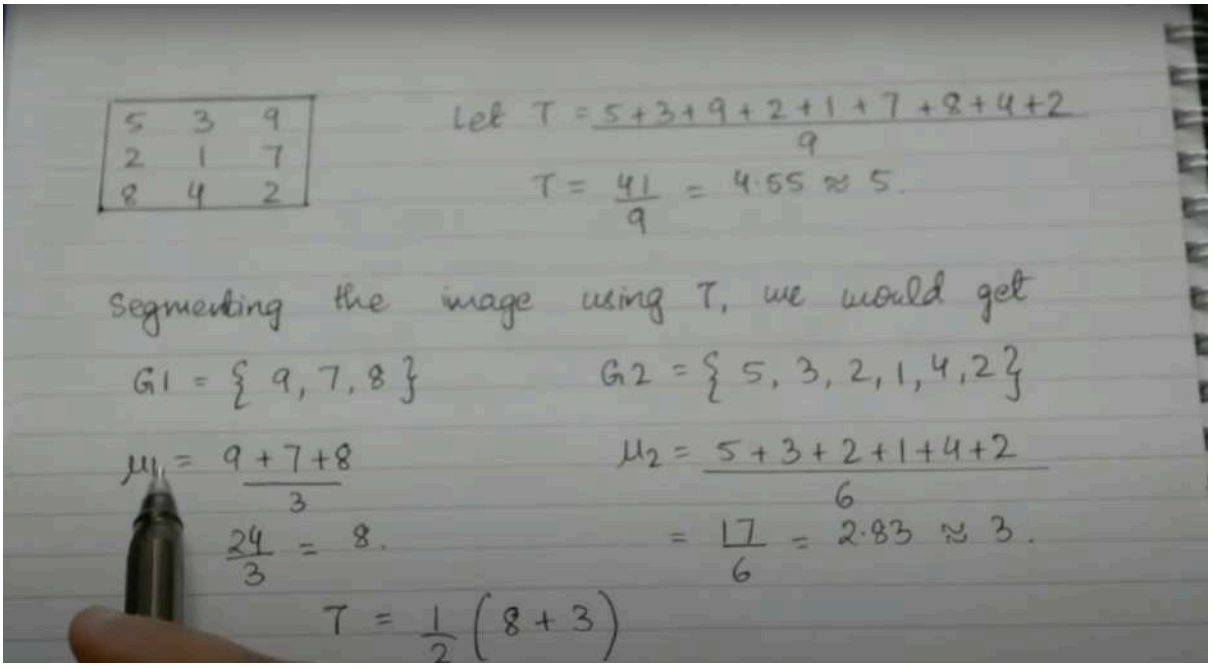
Procedure for Global Thresholding to obtain T

1) Select an initial estimate for T. (This value should be greater than the minimum and less than the maximum intensity level in the image. It is better to choose the average intensity of an image.)

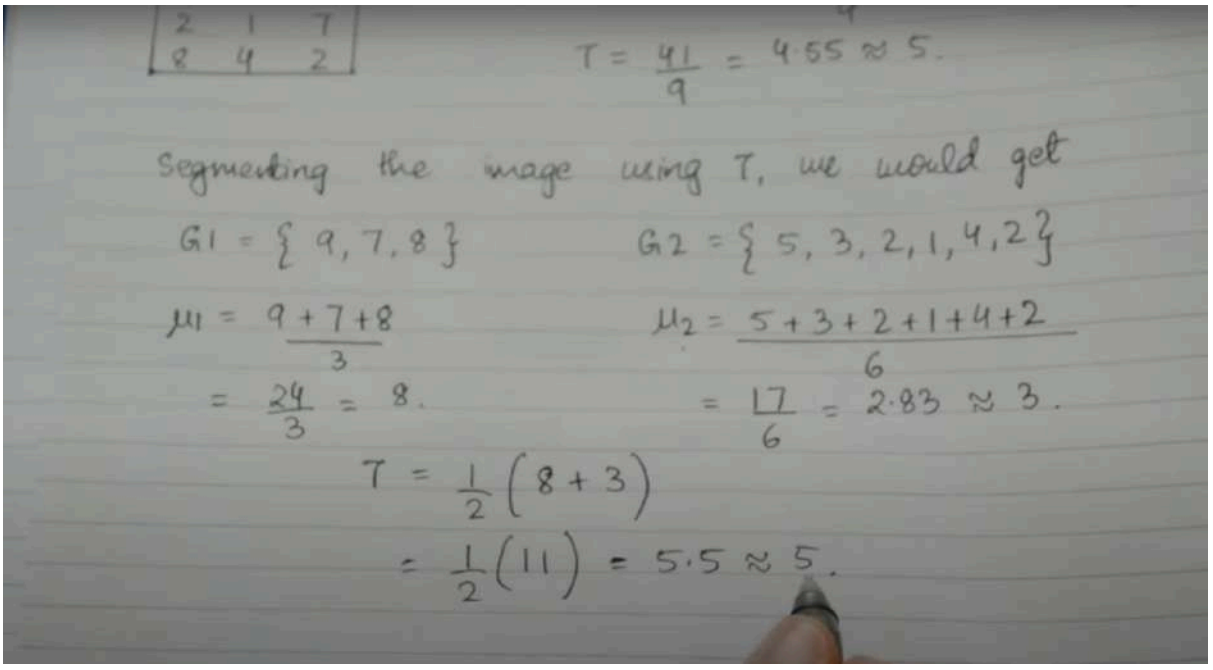
2) Segment the image using T. This will produce 2 groups of pixels : G1 consisting of all pixels with gray level values > T and G2 consisting of pixels with values <= T.

3) Compute the average gray level values $\mu 1$ and $\mu 2$ for the pixels in regions G1 and G2.



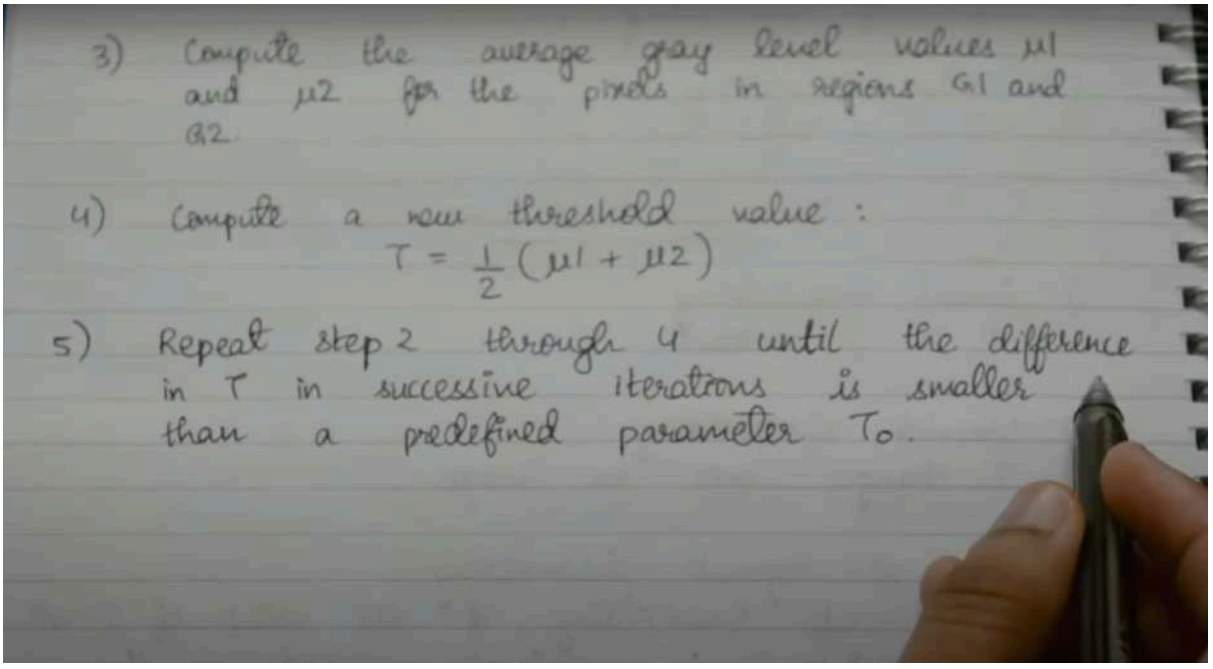


Average value as the thresholding, then separating into two G1 and G2 such that mui1 and mui2 and we get



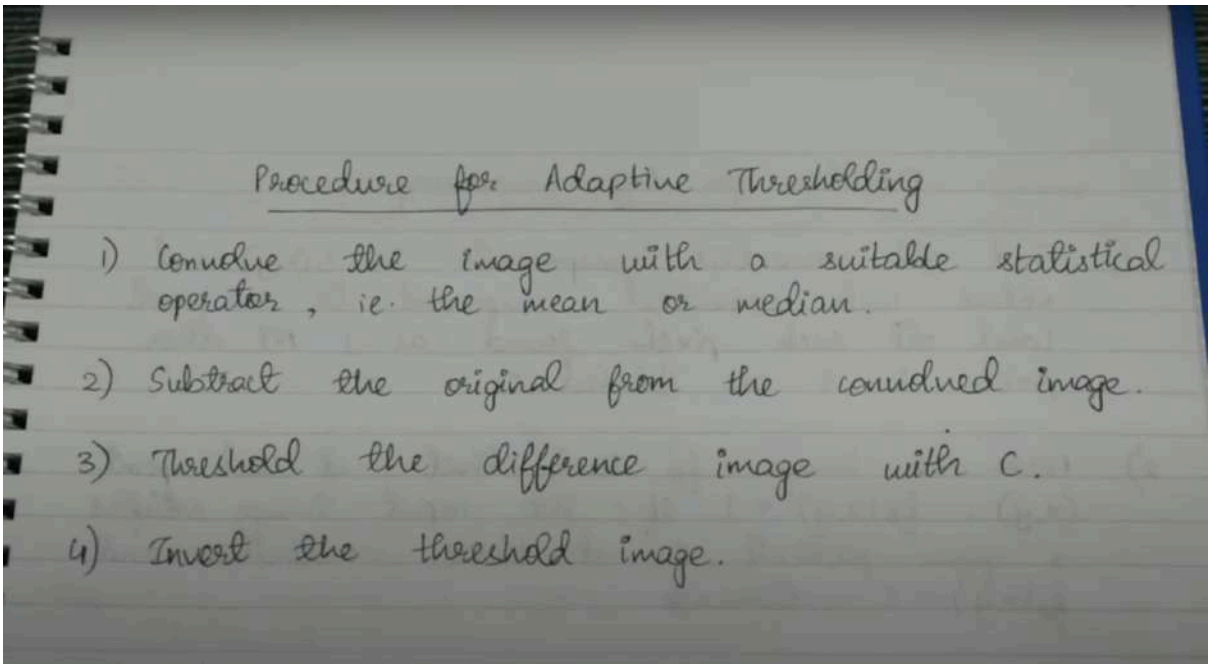
Since the new threshold and and old thresholds are similar and approximate, we can stop the process here.

Conditions applied if a similar value is not found until:



Local Thresholding

Global Thresholding uses two dominant modes/regions One darker and one bright region but local thresholding has more than one darker and bright region. Local or adaptive has more than one T.



Example:

Consider a 3x3 grayscale image matrix:

Step 1: Original Image Matrix

$$\begin{bmatrix} 5 & 7 & 6 \\ 4 & 5 & 8 \\ 3 & 4 & 7 \end{bmatrix}$$

Step 2: Convolve the Image with Mean Filter

Using a 3x3 mean filter (average of all values in the matrix):

$$\text{Mean} = \frac{5+7+6+4+5+8+3+4+7}{9} = \frac{49}{9} \approx 5$$

Convolving the image yields a matrix with each element replaced by the mean:

$$\begin{bmatrix} 5 & 5 & 5 \\ 5 & 5 & 5 \\ 5 & 5 & 5 \end{bmatrix}$$

Step 3: Subtract the Original from the Convolved Image

Subtract the convolved image from the original:

$$\begin{bmatrix} 5-5 & 7-5 & 6-5 \\ 4-5 & 5-5 & 8-5 \\ 3-5 & 4-5 & 7-5 \end{bmatrix} = \begin{bmatrix} 0 & 2 & 1 \\ -1 & 0 & 3 \\ -2 & -1 & 2 \end{bmatrix}$$

Step 4: Threshold the Difference Image with C

Set threshold `C = 1`. Values greater than `C` are set to 1, and others are set to 0:

$$\begin{bmatrix} 0 & 2 & 1 \\ -1 & 0 & 3 \\ -2 & -1 & 2 \end{bmatrix} \rightarrow \begin{bmatrix} 0 & 1 & 1 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \end{bmatrix}$$

Step 3: Subtract the Original from the Convolved Image

Subtract the convolved image from the original:

$$\begin{bmatrix} 5-5 & 7-5 & 6-5 \\ 4-5 & 5-5 & 8-5 \\ 3-5 & 4-5 & 7-5 \end{bmatrix} = \begin{bmatrix} 0 & 2 & 1 \\ -1 & 0 & 3 \\ -2 & -1 & 2 \end{bmatrix}$$

Step 4: Threshold the Difference Image with C

Set threshold `C = 1`. Values greater than `C` are set to 1, and others are set to 0:

$$\begin{bmatrix} 0 & 2 & 1 \\ -1 & 0 & 3 \\ -2 & -1 & 2 \end{bmatrix} \rightarrow \begin{bmatrix} 0 & 1 & 1 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \end{bmatrix}$$

Step 5: Invert the Thresholded Image

Invert the thresholded image:

$$\begin{bmatrix} 0 & 1 & 1 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \end{bmatrix} \rightarrow \begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ 1 & 1 & 0 \end{bmatrix}$$

Summary:

- **Convolution** smooths the image using the mean value.
- **Subtraction** highlights differences between the original and smoothed images.
- **Thresholding** filters significant changes, and **inversion** emphasizes these areas for clearer results.

This simplified approach demonstrates how adaptive thresholding can adjust locally, especially useful in uneven lighting conditions or varying contrasts within an image.

Explanation:

- **0 becomes 1:** This step makes previously ignored areas (in the thresholding step) visible.
- **1 becomes 0:** This highlights the significant areas identified during thresholding.

Adaptive Thresholding

Example:

Step 1: Consider a small 3x3 grayscale image

We have a 3x3 pixel grayscale image, where pixel intensity values range from 0 to 255:

$$\begin{bmatrix} 120 & 125 & 130 \\ 110 & 115 & 120 \\ 100 & 105 & 110 \end{bmatrix}$$

Step 2: Choose a window size

Since the matrix is small, we'll use the entire **3x3 window** to compute the threshold for each pixel.

Step 3: Compute the mean of the window

The mean of the entire 3x3 matrix is calculated as:

$$\text{Mean} = \frac{120 + 125 + 130 + 110 + 115 + 120 + 100 + 105 + 110}{9} = \frac{1035}{9} \approx 115$$

Step 4: Subtract a constant

We subtract a constant (let's say **C = 5**) from the mean to get the threshold value:

$$\text{Threshold} = 115 - 5 = 110$$

Step 5: Apply the threshold

Now, we apply this threshold to each pixel in the image. If the pixel intensity is greater than the threshold (110), it becomes **255 (white)**; otherwise, it becomes **0 (black)**.

1. For pixel **120**, since $120 > 110$, set it to **255**.
2. For pixel **125**, since $125 > 110$, set it to **255**.
3. For pixel **130**, since $130 > 110$, set it to **255**.
4. For pixel **110**, since $110 = 110$, it remains **0**.
5. For pixel **115**, since $115 > 110$, set it to **255**.
6. For pixel **120**, since $120 > 110$, set it to **255**.
7. For pixel **100**, since $100 < 110$, set it to **0**.
8. For pixel **105**, since $105 < 110$, set it to **0**.
9. For pixel **110**, since $110 = 110$, it remains **0**.

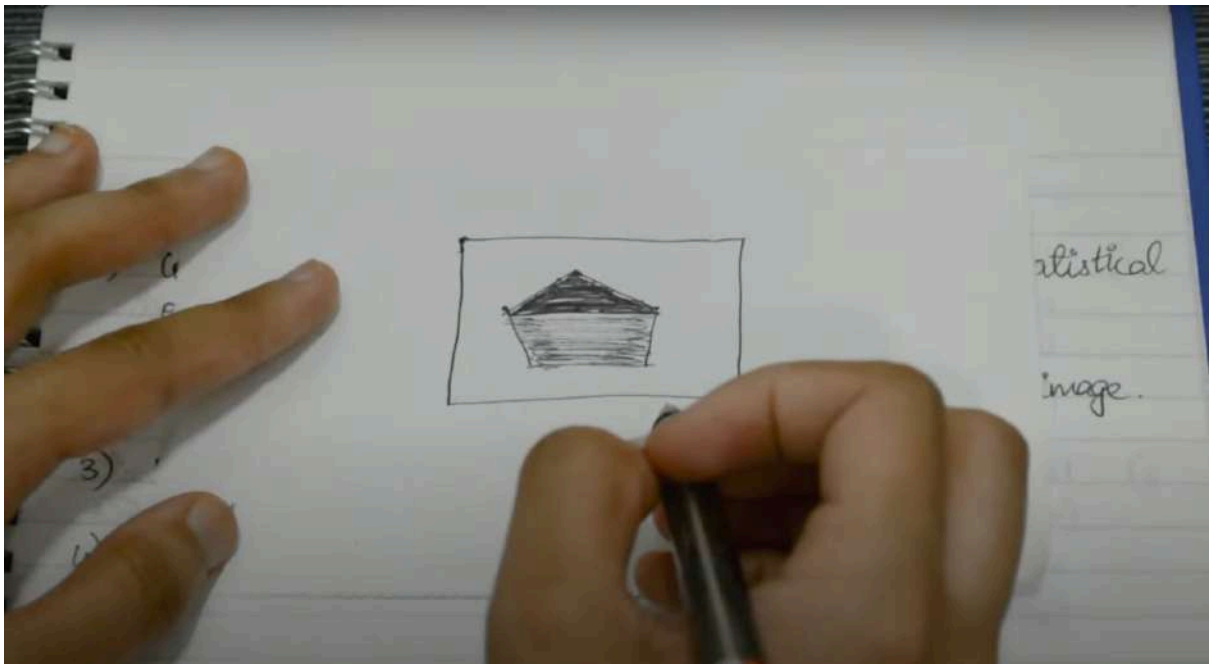
Step 6: Resulting thresholded image

After applying the adaptive thresholding, the binary image becomes:

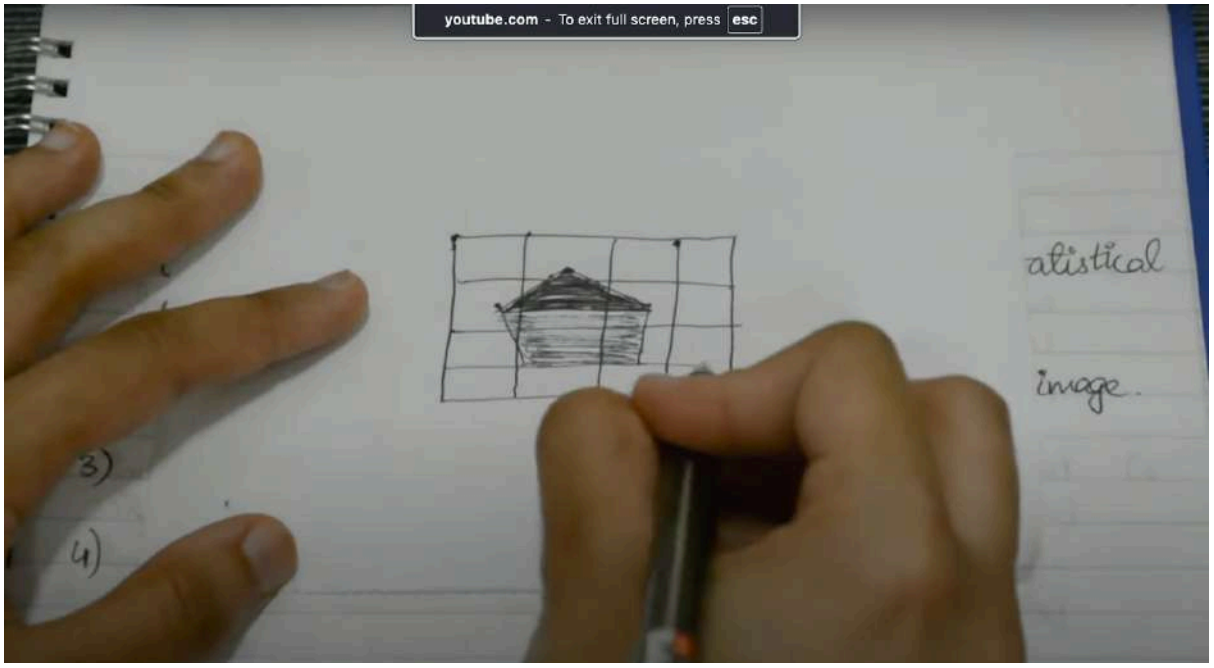
$$\begin{bmatrix} 255 & 255 & 255 \\ 0 & 255 & 255 \\ 0 & 0 & 0 \end{bmatrix}$$

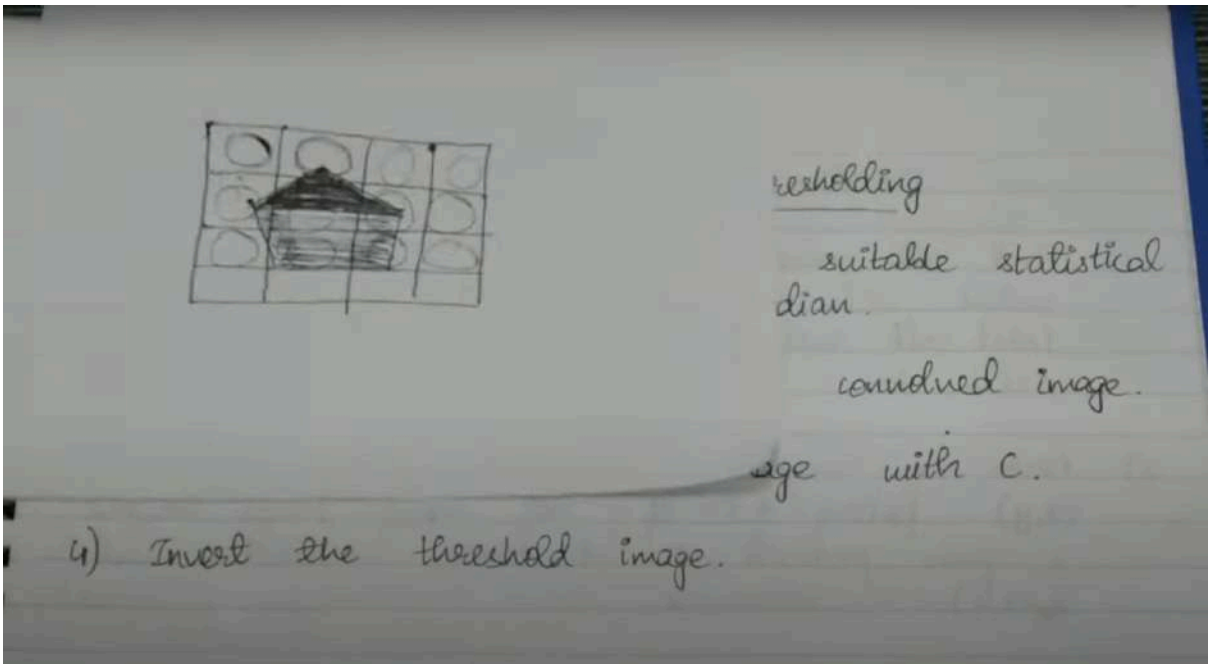
Explanation:

- We computed the **mean** of the whole image as 115 and used a constant subtraction of 5 to determine the threshold (110).
- Pixels above the threshold become white (255), while pixels at or below the threshold become black (0).

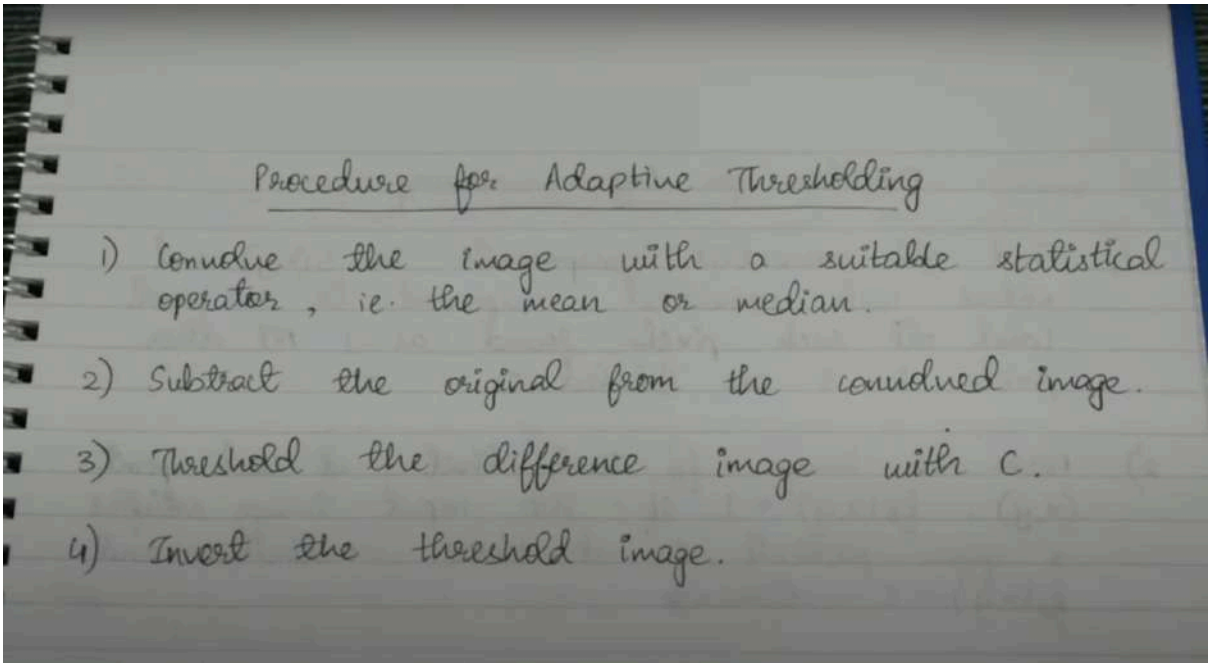


Divide into the blocks and do the global thresholding for each of these blocks.





Adaptive Thresholding



Morphological Operations

Morphological Operators

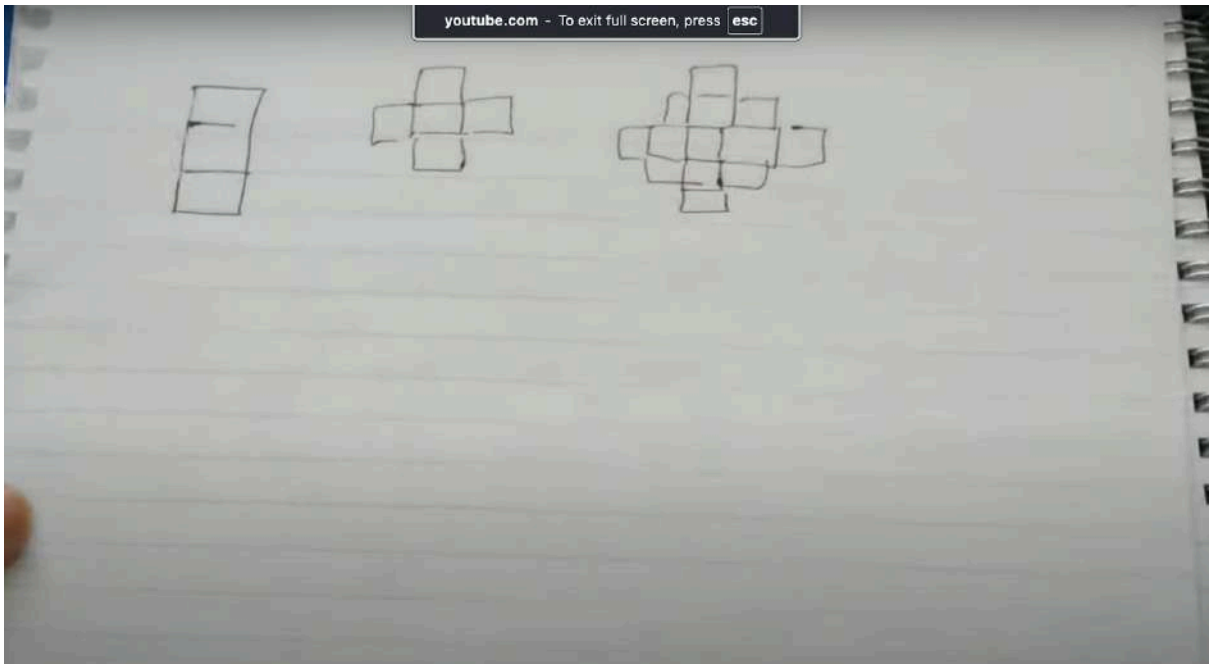
Morphological image processing (or morphology) describes a range of image processing techniques that deal with the shape (or morphology) of features in an image.

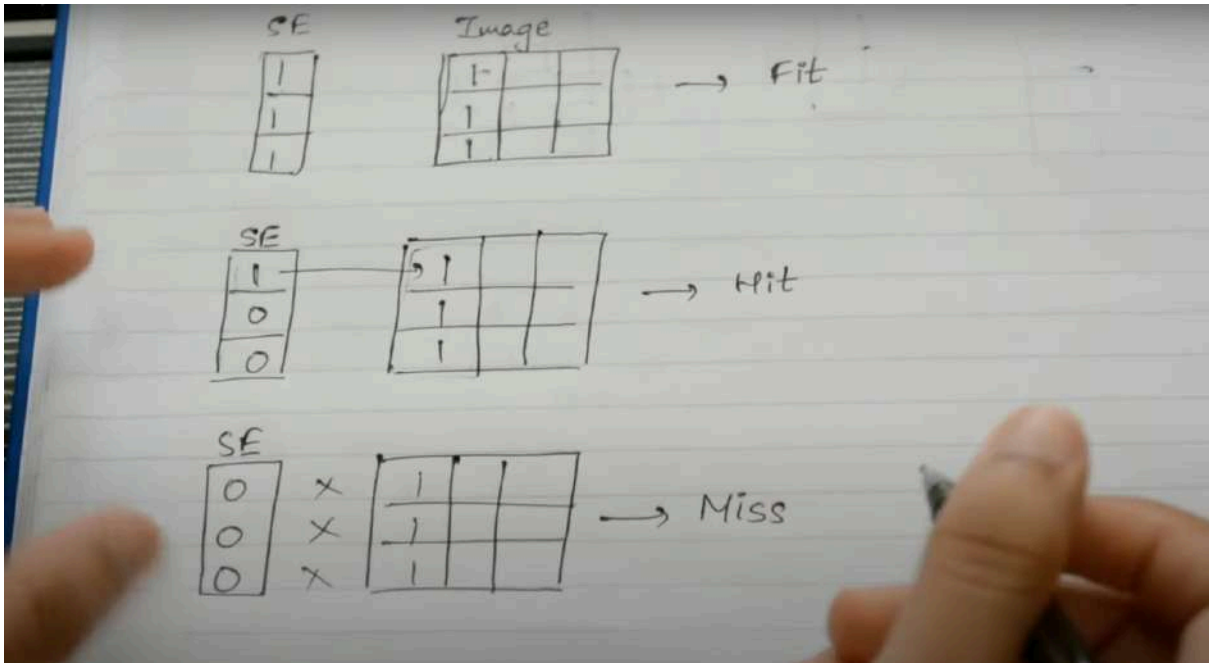
Morphological operations are typically applied to remove imperfections introduced during segmentation, and so typically operate on bi-level images.

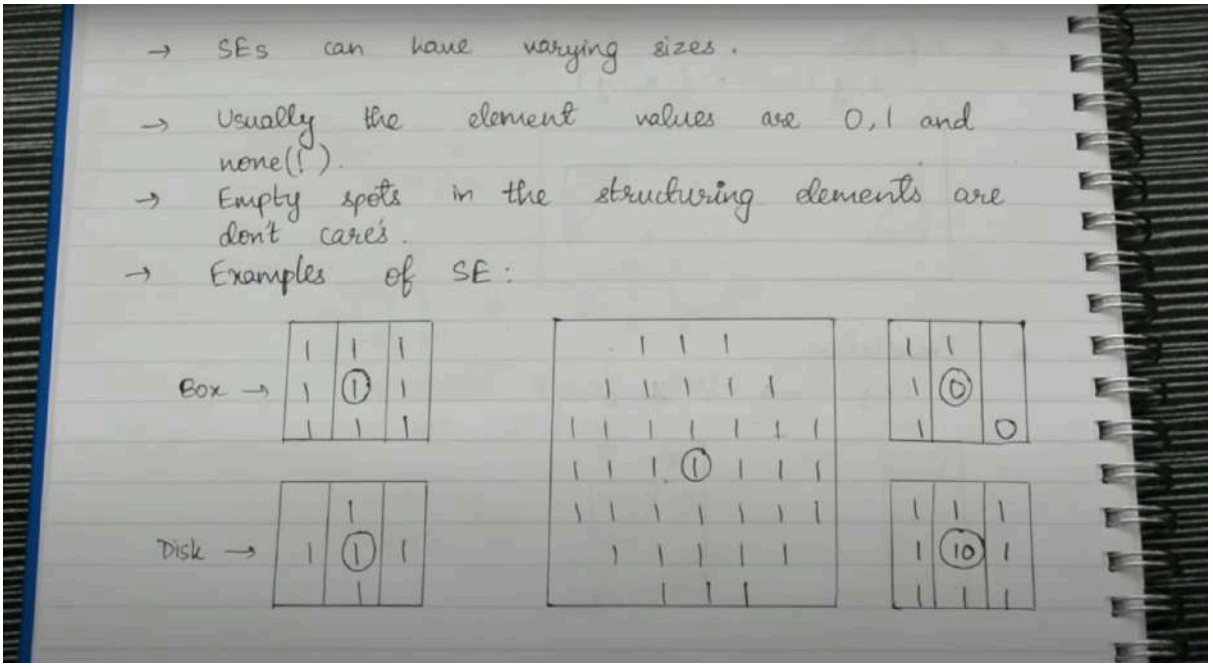
These operations are performed using the concepts of reflection and translation.

* Structuring Element (SE)

→ It is a small set to probe the image under study.
→ For each structuring element, we should define the origin.
→ The shape and size must be adapted to geometric properties for the object
→ We check for three conditions while applying the SE:
  - Fit
  - Hit
  - Miss


youtube.com - To exit full screen, press esc


SE
Image
Fit
SE
Hit
SE
Miss


→ SEs can have varying sizes.
→ Usually the element values are 0, 1 and none(!).
→ Empty spots in the structuring elements are don't cares.
→ Examples of SE:
Box →
Disk →

Question

Q1. Compute $A \oplus B$ for the following input images.

0	0	0	0	0	0
0	0	1	1	0	0
0	1	1	1	1	0
0	0	1	1	0	0
0	0	0	0	0	0

A

1
1
1

B

Ans. First we will pad the image A at bottom.

Ans. First we will pad the image A at top and bottom.

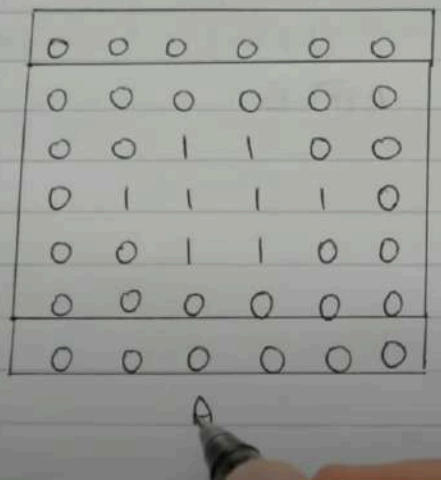
0	0	0	0	0	0
0	0	0	0	0	0
0	0	1	1	0	0
0	1	1	1	1	0
0	0	1	1	0	0
0	0	0	0	0	0
0	0	0	0	0	0

A

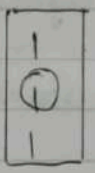
1
1
1

B

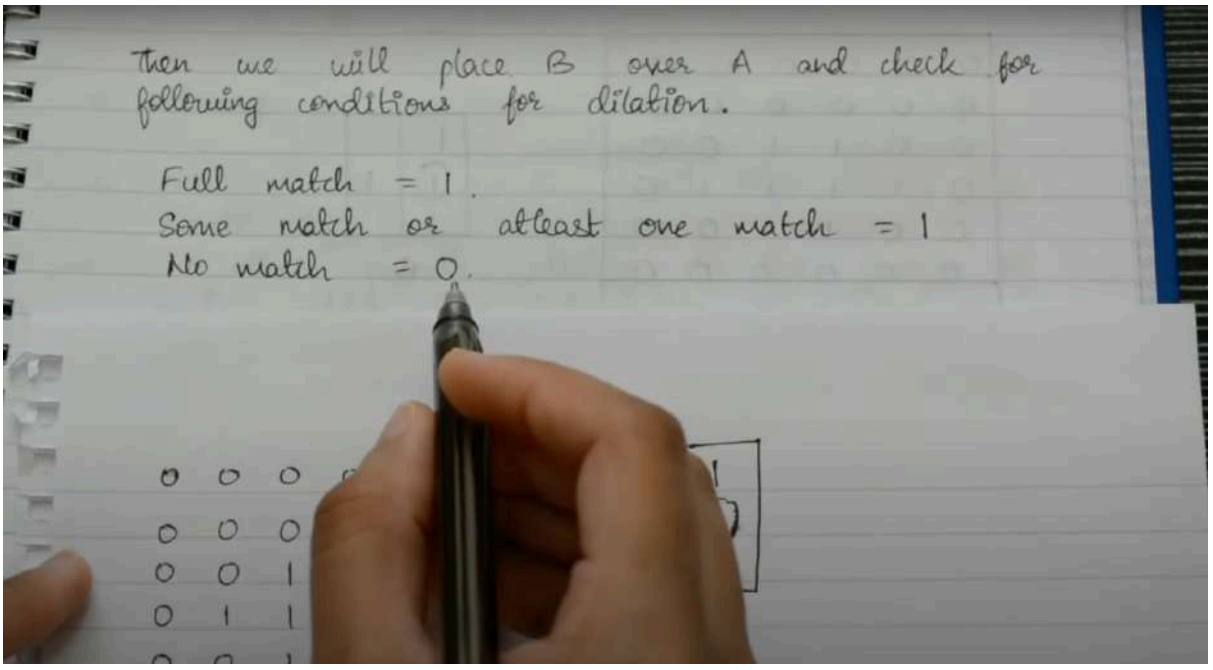
Ans. First we will pad the image A at top and bottom.

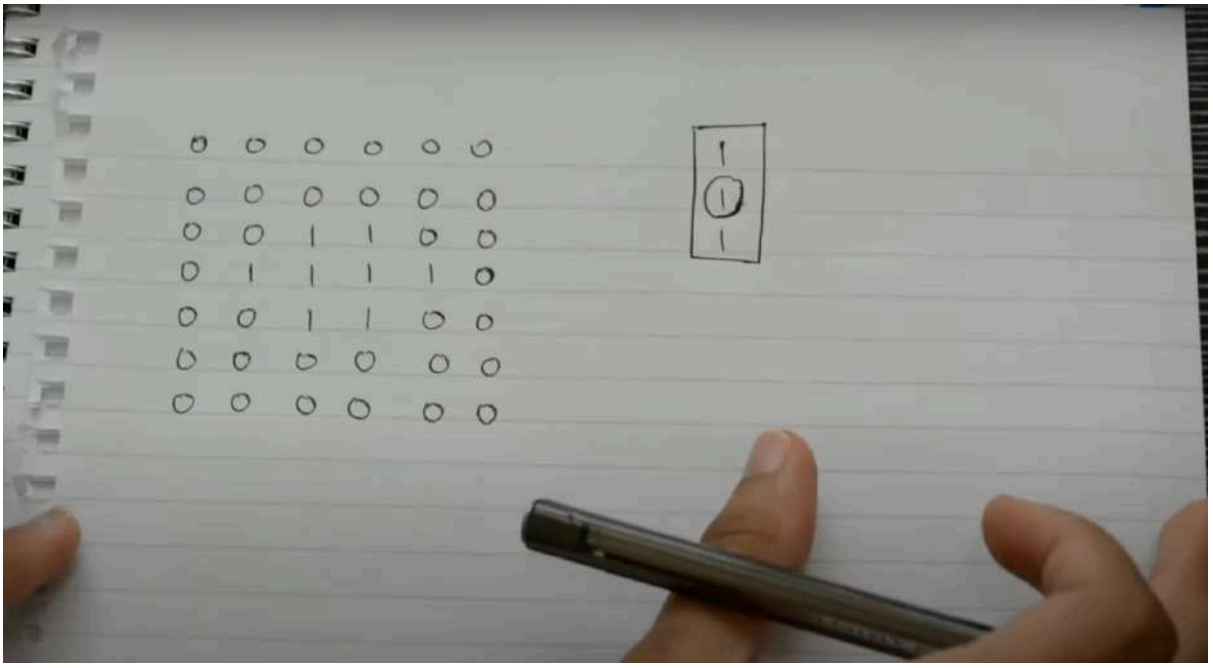


A



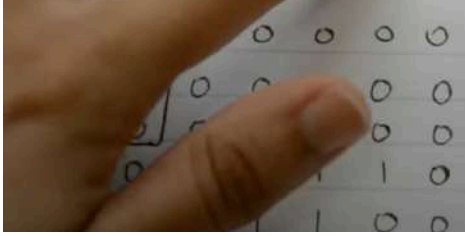
B

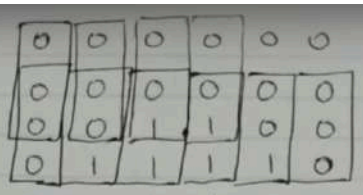

Then we will place B over A and check for following conditions for dilation.
Full match = 1.
Some match or atleast one match = 1
No match = 0.



Then we will place B over A and check for following conditions for dilation.

Full match = 1.
Some match or atleast one match = 1
No match = 0.





0 0 1 1 0 0
0 1 1 1 1 0

Then we will place B over A and check for following conditions for dilation.

Full match = 1.
Some match or atleast one match = 1
No match = 0.

Output image

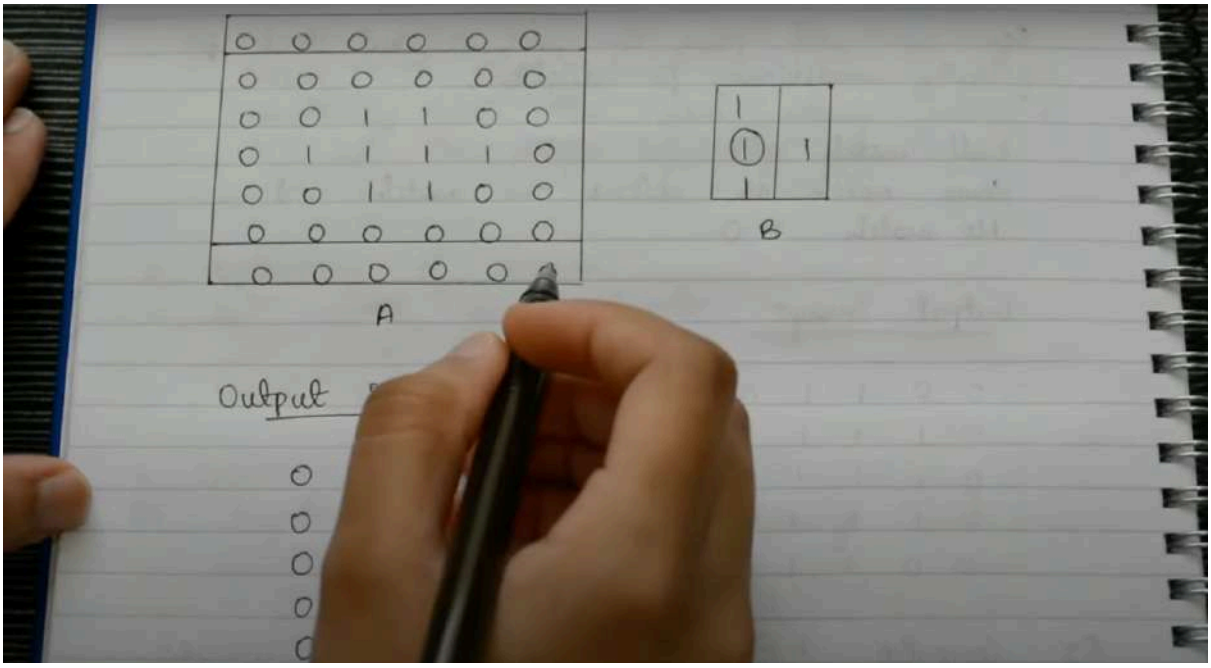
0	0	1	1	0	0
0	1	1	1	1	0
0	1	1	1	1	0
0	1	1	1	1	0
0	0	1	1	0	0

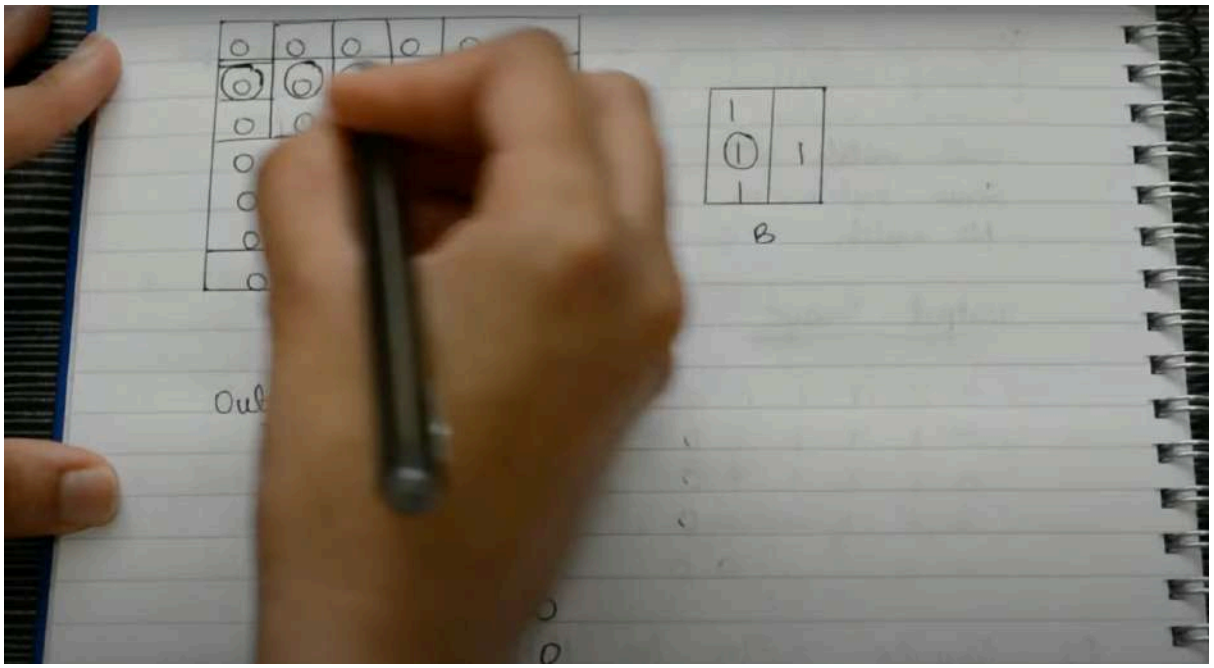
Then we will place B over A and check for following conditions for dilation.

Full match = 1.
Some match or atleast one match = 1
No match = 0.

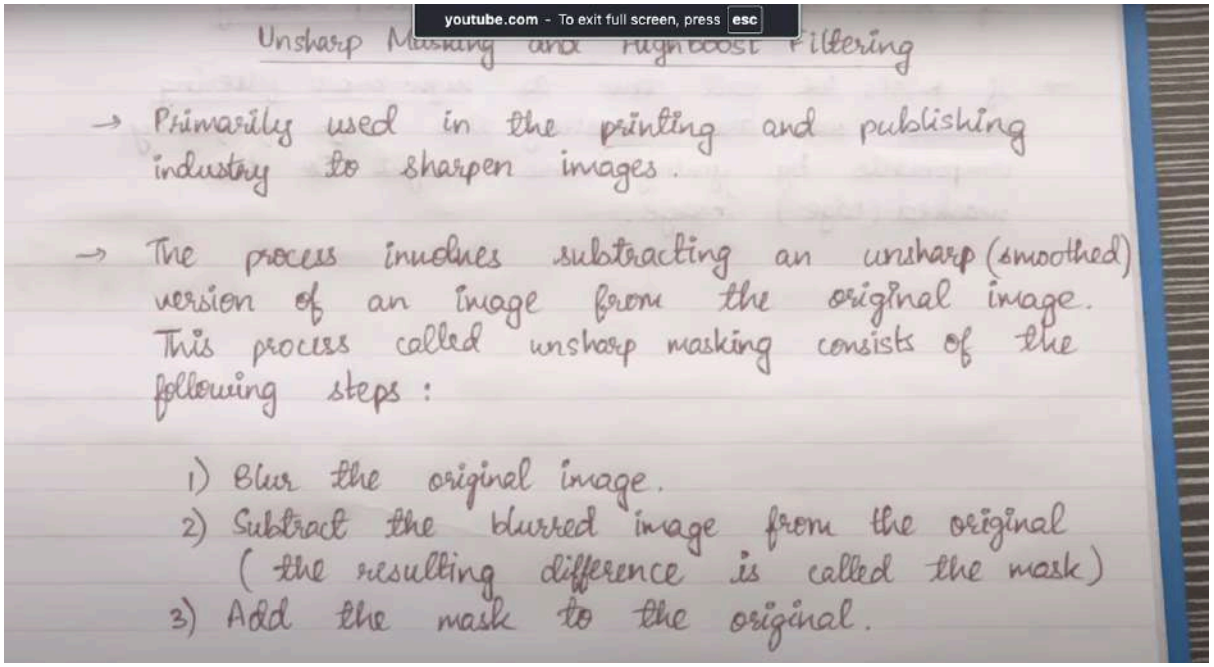
Output image

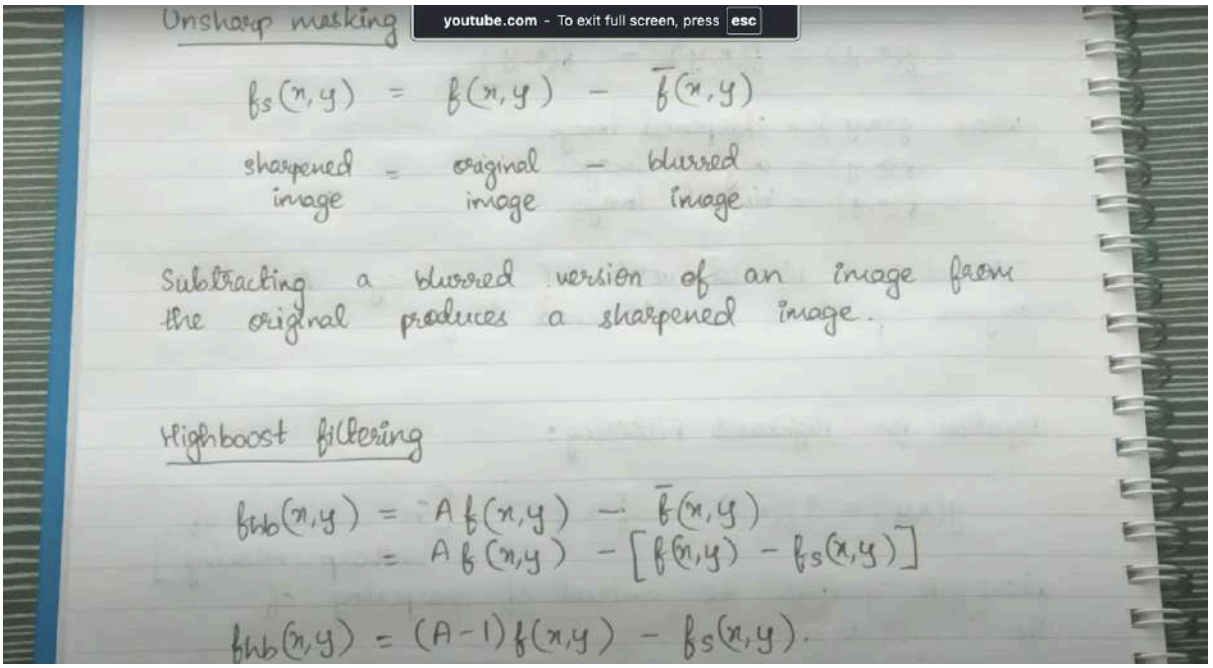
0	0	1	1	0	0
0	1	1	1	1	0
0	1	1	1	1	0
0	1	1	1	1	0
0	0	1	1	0	0



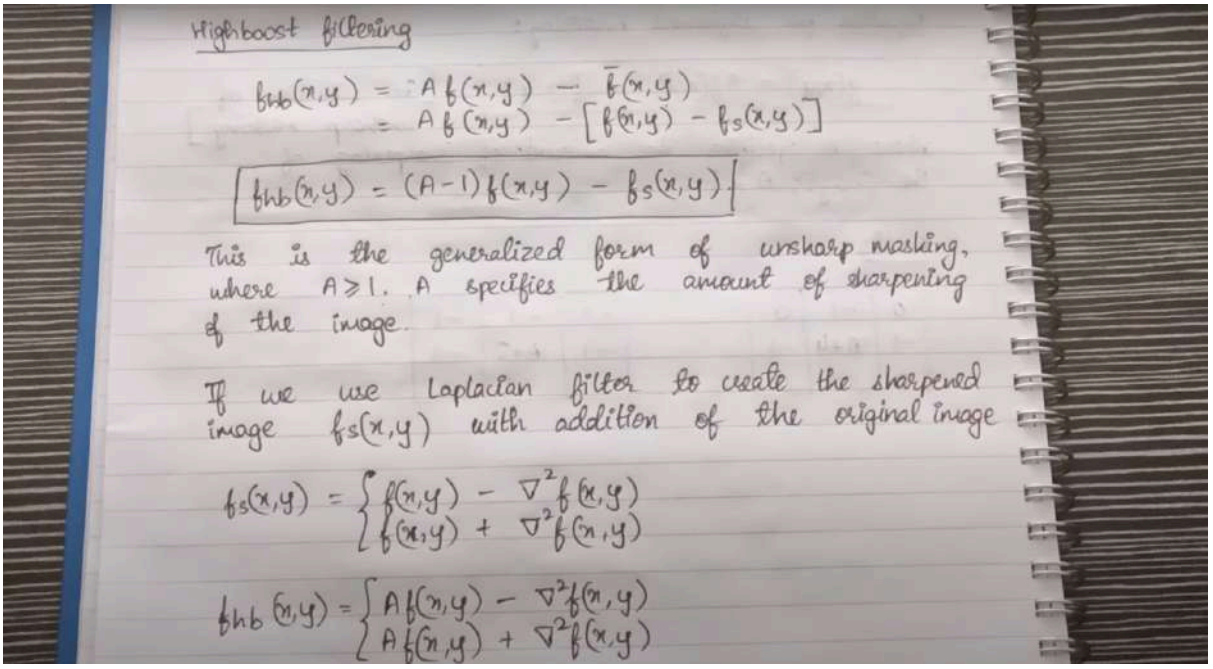


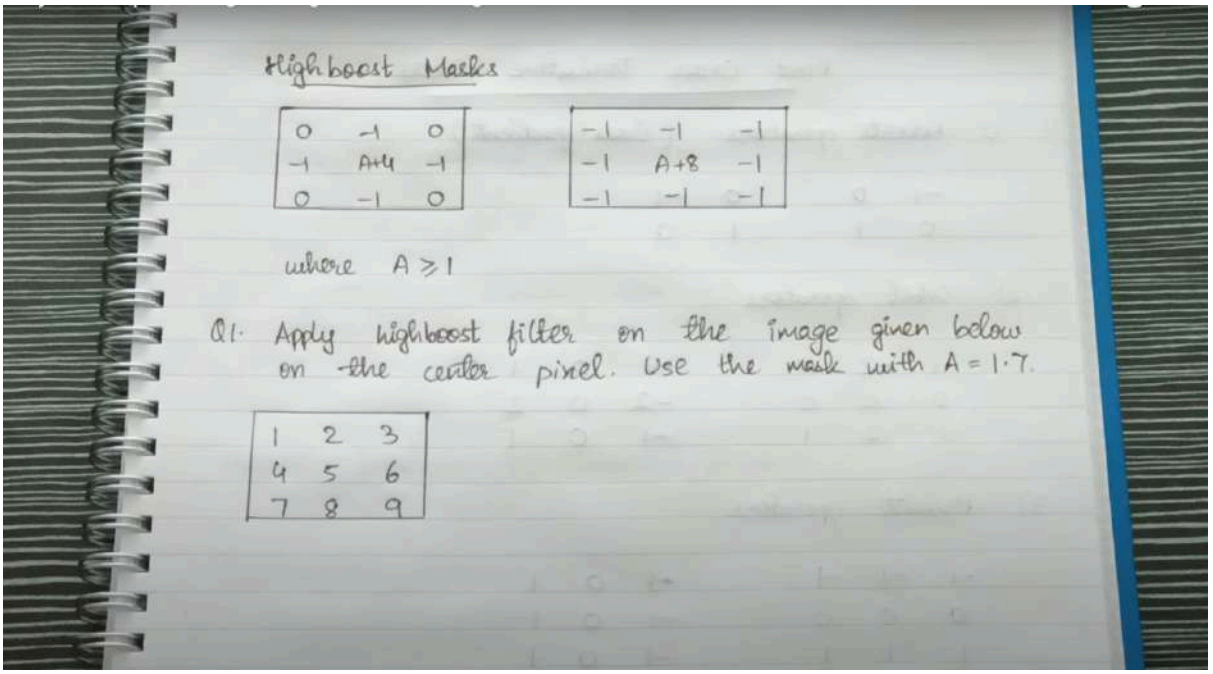
Unsharp Masking and High-boost Filtering





Keep f(x,y) + fs(x,y)





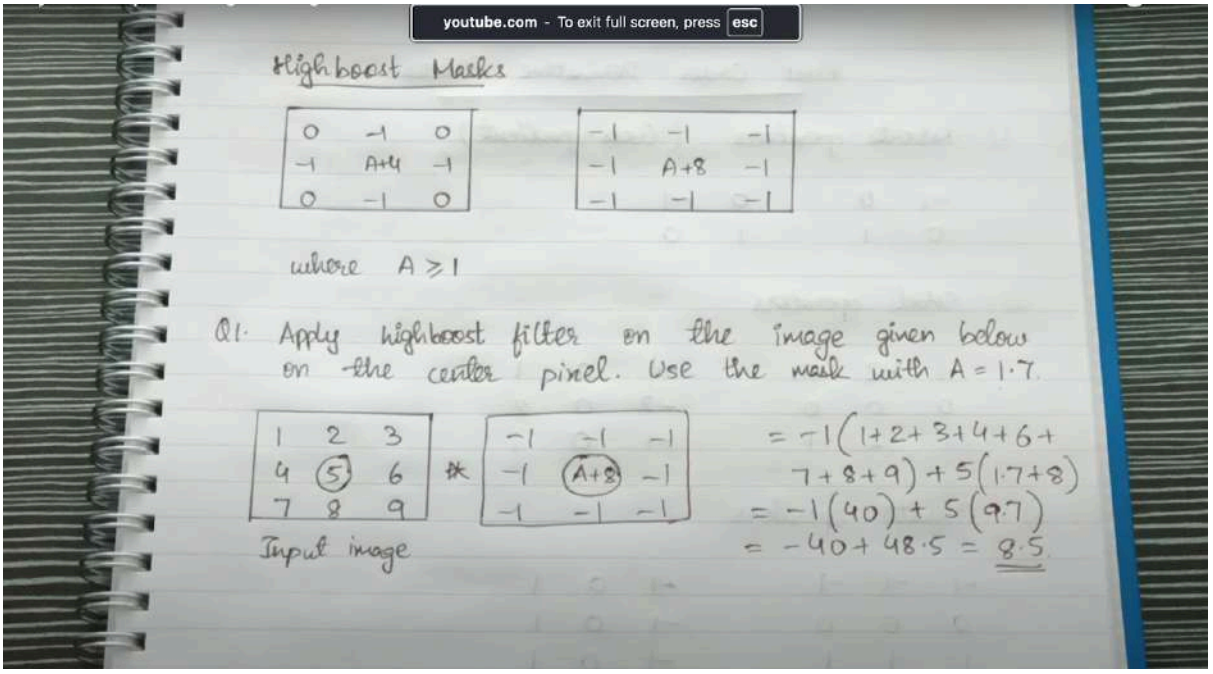
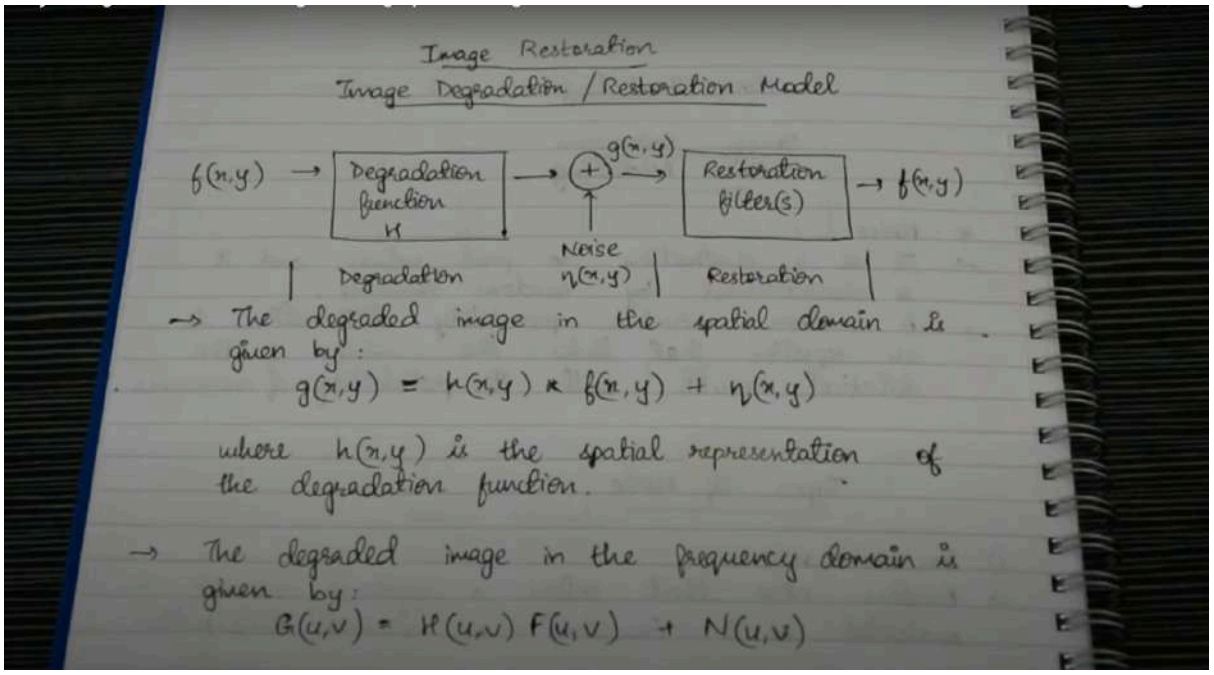
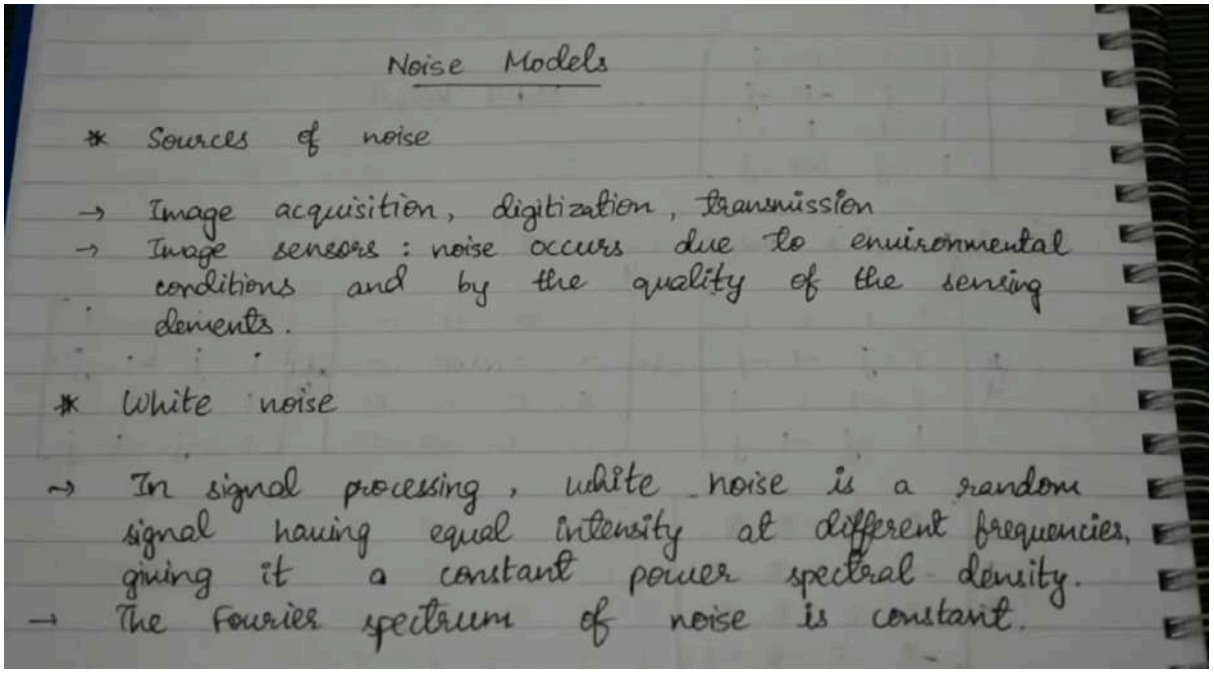


Image Restoration in digital image processing

- We also **focus** on the **noise present** in the **image unlike** in the **enhancement** we donot focus on the noise
- Attempt to recover the image in its best form

- If we have blurred the image we can remove that blurriness in the images by using prior knowledge of image degradation
- If the noise is **additive** then we use **spatial** but if the degradation is due to image blur then it is difficult to remove it with spatial filter so we use **frequency removal filter**





Types of Noise

1) Gaussian noise
→ Random noise that enters a system can be modelled as a Gaussian or normal distribution.

→ This noise affects both, dark and light areas of image.

→ The PDF of a Gaussian random variable, z, is given by

$$p(z) = \frac{1}{\sqrt{2\pi}\sigma} e^{-(z-\mu)^2/2\sigma^2}$$

where $z \rightarrow$ gray level
$\mu \rightarrow$ mean of average values of z
$\sigma \rightarrow$ standard deviation
$\sigma^2 \rightarrow$ variance

2) Impulse noise
→ It is also known as shot noise, salt and pepper noise, and binary noise.

→ It occurs mostly because of sensor and memory problem because of which pixels are assigned incorrect maximum values

→ The PDF of (bipolar) impulse noise is given by

$$p(z) = \begin{cases} P_a & \text{for } z = a \\ P_b & \text{for } z = b \\ 0 & \text{otherwise} \end{cases}$$

If either P_a or P_b is zero, impulse noise is called unipolar.

Q1. Why is impulse noise known as salt and pepper noise?

Ans. If neither of the two probabilities P_a or P_b are zero, and especially if they are approximately equal, impulse noise values resemble salt- and -pepper granules randomly distributed over the image. For this reason, bipolar impulse noise is also called salt-and-pepper noise.

3) Poisson noise

→ This type of noise manifests as a random structure or texture in images.

3) Poisson noise

→ This type of noise manifests as a random structure or texture in images.

→ It is very common in X-ray images.

→ The PDF of Poisson noise is given by:

$$p(z) = \frac{(np)^z e^{-np}}{z!}$$

where n → total no. of pixels

z → gray level

p → ratio of noise pixels to the total no. of pixels

where $n \rightarrow$ total no. of pixels
$z \rightarrow$ gray level
$p \rightarrow$ ratio of noise pixels to the total no. of pixels

4) Exponential noise
→ This type of noise occurs mostly due to the illumination problems
It is observed in laser imaging

The PDF of exponential noise is given by:

$$p(z) = \begin{cases} ae^{-az} & \text{for } z \geq 0 \\ \text{otherwise} \end{cases}$$ where $a > 0$.

Mean → $1/a$
Variance → $1/a^2$

5) Gamma noise (Erlang)
→ This type of noise also occurs mostly due to the illumination problems.

→ The PDF is given by:

$$p(z) = \begin{cases} \frac{a^b z^{b-1}}{(b-1)!} e^{-az} & \text{for } z \geq 0 \\ 0 & \text{for } z < 0 \end{cases}$$

where $a > 0$ and b is a positive integer.

Mean → b/a

Variance → b/a^2

$a \rightarrow$ min. gray value
$b \rightarrow$ max. gray value

6) Rayleigh noise

→ This type of noise is mostly present in range images.

→ Range images are mostly used in remote sensing applications.

→ The PDF is given by:

$$p(z) = \begin{cases} \frac{2}{b}(z-a)e^{-\frac{(z-a)^2}{b}} & \text{for } z \geq a \\ 0 & \text{otherwise} \end{cases}$$

Mean → $a + \sqrt{\frac{\pi b}{4}}$

Variance → $\frac{b(4-\pi)}{4}$

7) Uniform noise

→ It is also a very popular noise which occurs in images where different values of noise are equally probable.

→ It occurs because of Quantization noise.

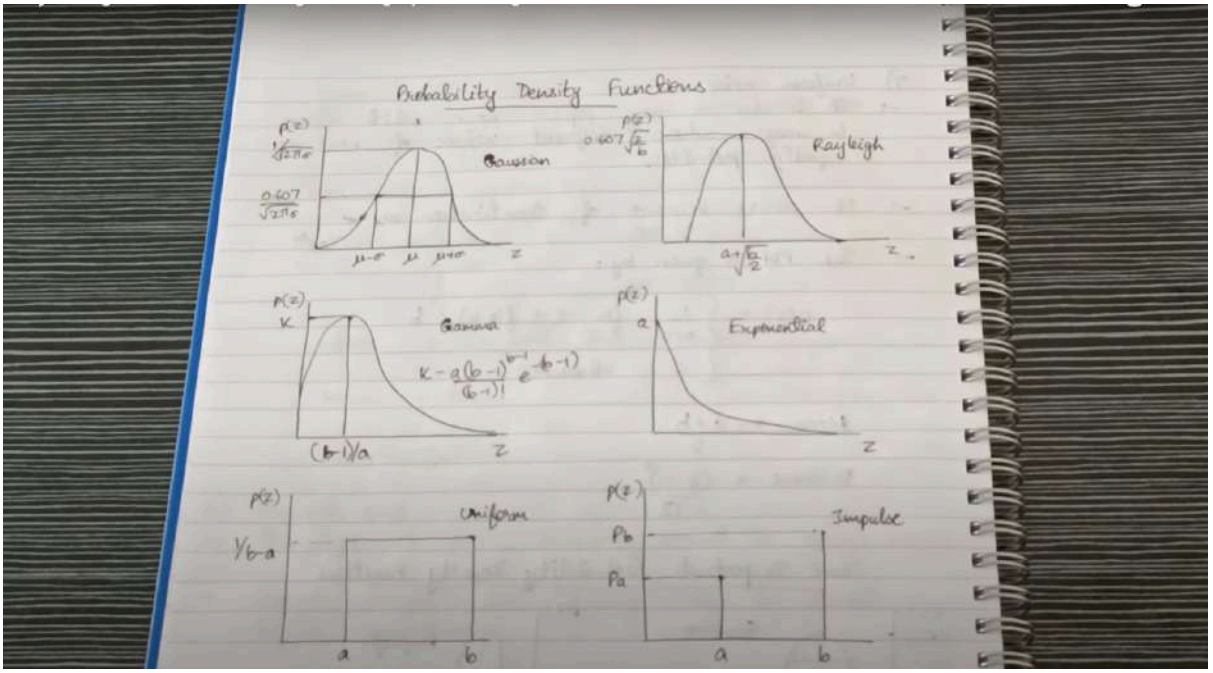
→ The PDF is given by:

$$p(z) = \begin{cases} \frac{1}{b-a} & \text{for } a \leq f(x,y) \leq b \\ 0 & \text{otherwise} \end{cases}$$

Mean → $\frac{a+b}{2}$

Variance → $\frac{(b-a)^2}{12}$

8) Periodic Noise
→ Arises typically from electrical or electromechanical interference.
→ Reduced significantly via frequency domain filtering.



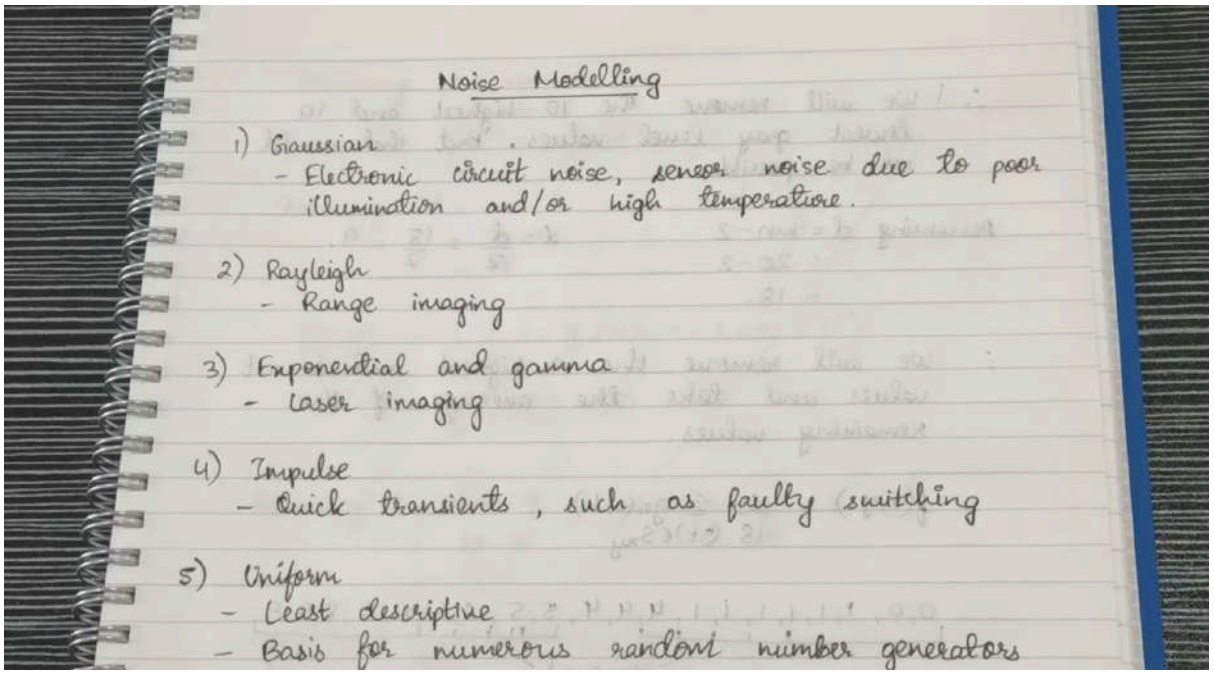
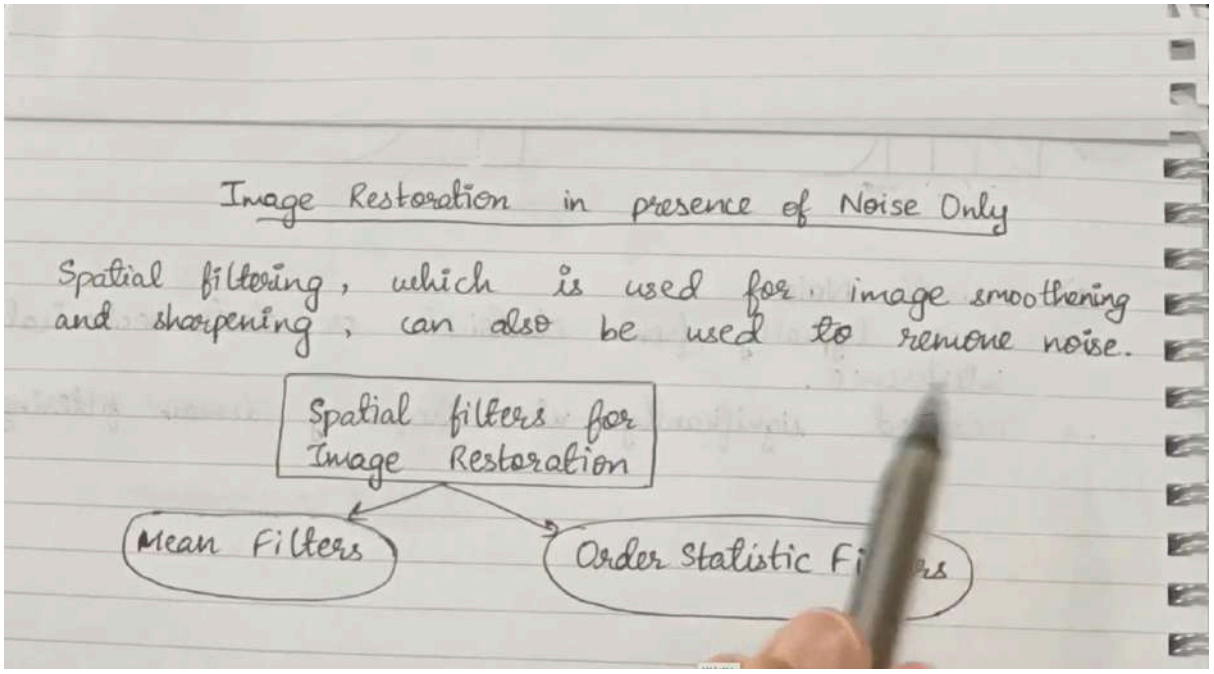


Image Restoration in the Presence of Noise Only



Mean Filters

1) Arithmetic mean filter

→ This filter removes local variations within the image.

→ It is similar to low pass filter.

→ It is useful in removing Gaussian noise and uniform noise.

$$f(x,y) = \frac{1}{mn} \sum_{(s,t) \in S_{xy}} g(s,t)$$

where $g(s,t)$ → mask applied over blurred image

2) Geometric mean filter

→ This filter eliminates Gaussian noise.

→ It is ineffective for pepper type of noise.

→ This filter achieves smoothing comparable to the arithmetic mean filter, but it tends to lose less image details in the process.

$$f(x,y) = \left[\prod_{(s,t) \in S_{xy}} g(s,t) \right]^{1/mn}$$

2) Geometric mean filter

→ This filter eliminates Gaussian noise.

→ It is ineffective for pepper type of noise.

→ This filter achieves smoothing comparable to the arithmetic mean filter, but it tends to lose less image details in the process.

$$f(x,y) = \left[\prod_{(s,t)\in S_{xy}} g(s,t)\right]^{1/mn}$$

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}_{3\times 3}$$

$$= (1\times 2\times 3\times 4\times 5\times 6\times 7\times 8\times 9)^{1/9}$$

4) Contra-harmonic mean filter

→ It is well suited for reducing the effects of salt-and-pepper noise.

→ $Q > 0$ for elimination of pepper noise and $Q < 0$ for elimination of salt noise.

→ Q is the order of the filter.

$$f(x,y) = \frac{\sum_{(s,t)\in S_{xy}} g(s,t)^{Q+1}}{\sum_{(s,t)\in S_{xy}} g(s,t)^{Q}}$$

when $Q = 0$, it becomes arithmetic mean filter.
$Q = 1$, it becomes harmonic mean filter.

* Usage :

→ Q is the order of the filter.

$$f(x,y) = \frac{\sum_{(s,t)\in S_{xy}} g(s,t)^{Q+1}}{\sum_{(s,t)\in S_{xy}} g(s,t)^{Q}}$$

when $Q = 0$, it becomes arithmetic mean filter.
$Q = 1$, it becomes harmonic mean filter.

* Usage :

→ Arithmetic and geometric mean filters : suited for Gaussian or uniform noise.
→ Contraharmonic filters : suited for impulse noise.

Order Statistic Filters

1) Median filter.

→ It is an example of non-linear filter.
→ It works by sorting the list and finding the median. The center pixel is then replaced by the median value.
→ It is effective in the presence of both bipolar and unipolar impulse noise.

$$f(x,y) = \underset{(s,t)\in S_{xy}}{\text{median}}\{g(s,t)\}$$

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & (5) & 6 \\ 7 & 8 & 9 \end{bmatrix}$$

2) Maximum filter

→ It selects the largest value in the sorted list.
→ It is used for removing pepper noise

$$f(x,y) = \max_{(s,t)\in S_{xy}} \{g(s,t)\}$$

3) Minimum filter

→ It selects the minimum value in the sorted list.
→ It is used for removing salt noise.

$$f(x,y) = \min_{(s,t)\in S_{xy}} \{g(s,t)\}$$

4) Midpoint filter

→ This filter selects the midpoint, which is the average of the minimum and maximum values.
→ It is very effective in removing Gaussian noise and Uniform noise.

$$f(x,y) = \frac{1}{2}\left[\max_{(s,t)\in S_{xy}} \{g(s,t)\} + \min_{(s,t)\in S_{xy}} \{g(s,t)\}\right]$$

5) Alpha-trimmed mean filter

5) Alpha-trimmed mean filter

→ It is based on the concept of compilation of the average of the pixels that falls within the window.

→ It works by deleting the d/2 lowest and the d/2 highest gray-level values.

→ It is useful in situations involving multiple types of noise, such as a combination of salt-and-pepper and Gaussian noise.

$$f(x,y) = \frac{1}{mn-d} \sum_{(s,t)\in S_{xy}} g_r(s,t)$$

[If $d=0 \rightarrow$ arithmetic mean filter]

[If $d = mn-1 \rightarrow$ median filter]

6) Trimmed mean filter

d/2 highest gray-level values.

→ It is useful in situations involving multiple types of noise, such as a combination of salt-and-pepper and Gaussian noise.

$$f(x,y) = \frac{1}{mn-d} \sum_{(s,t)\in S_{xy}} g_r(s,t)$$

[If $d=0 \rightarrow$ arithmetic mean filter]

[If $d = mn-1 \rightarrow$ median filter]

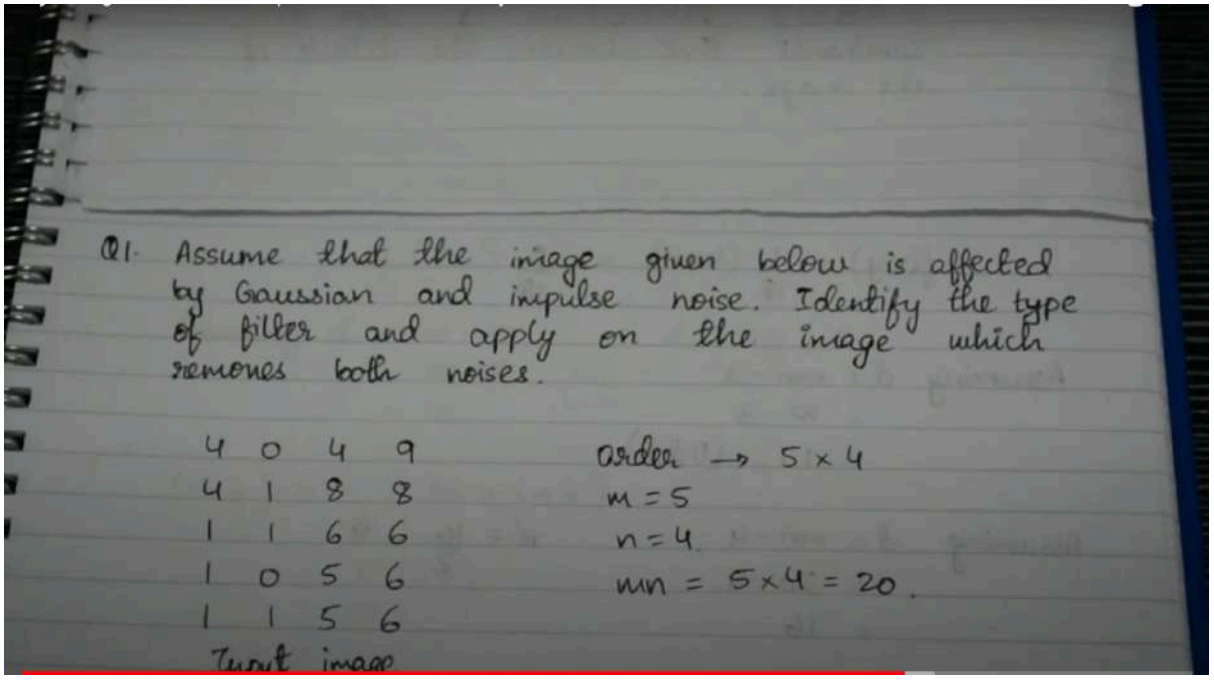
6) Trimmed mean filter

→ It works by removing the max. and min. values and calculating the average of the remaining values.

0, 2, 3, 4, 5, 6, 7, 8, 9

7

Question



Q1. Assume that the image given below is affected by Gaussian and impulse noise. Identify the type of filter and apply on the image which removes both noises.

4	0	4	9
4	1	8	8
1	1	6	6
1	0	5	6
1	1	5	6

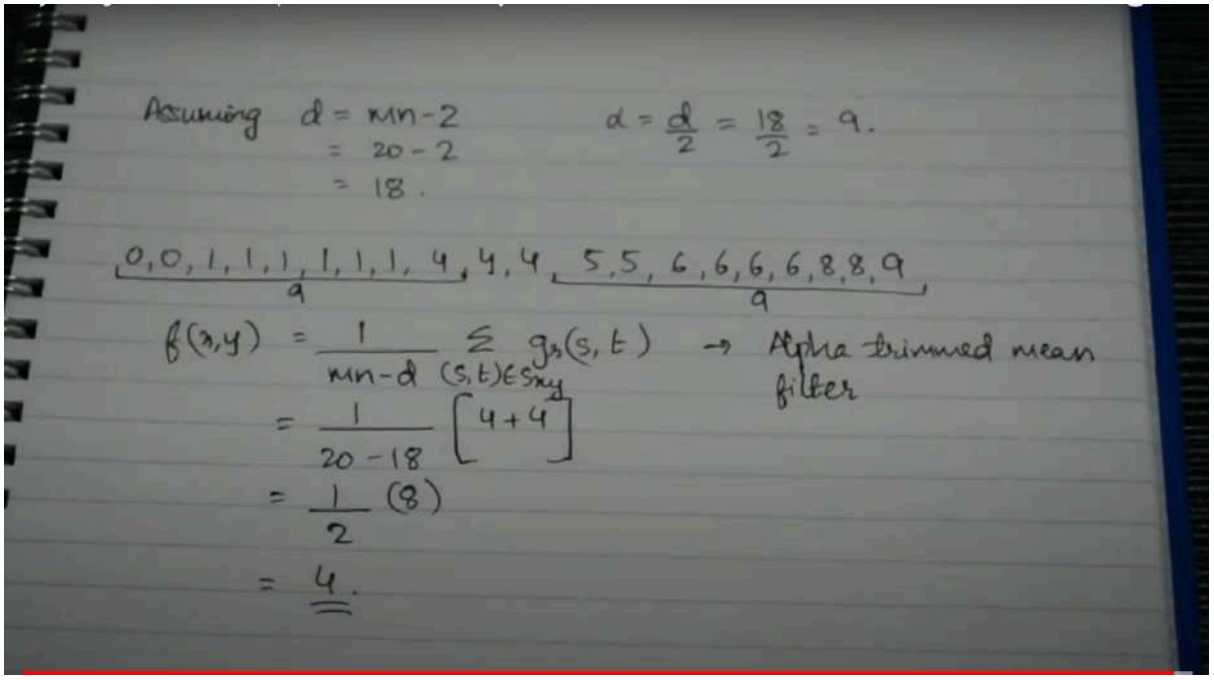
Input image

order → 5 × 4

$m = 5$

$n = 4$

$mn = 5 \times 4 = 20$.



Assuming $d = mn - 2$

$= 20 - 2$

$= 18$.

$\alpha = \frac{d}{2} = \frac{18}{2} = 9$.

$\underbrace{0,0,1,1,1,1,1,1,4}_{9}, 4, 4, \underbrace{5,5,6,6,6,6,8,8,9}_{9}$

$f(x,y) = \frac{1}{mn-d} \sum_{(s,t) \in S_{xy}} g_s(s,t)$ → Alpha trimmed mean filter

$= \frac{1}{20-18} [4+4]$

$= \frac{1}{2} (8)$

$= 4$.

Image Transforms

→ Image transforms are mathematical tools that help us to convert images from spatial domain to frequency domain.

→ Advantages for transforming images:
- It may isolate critical components of image pattern so that they are directly accessible for analysis.
- It may place image data in a more compact form so that it can be stored and transmitted efficiently.
- It is useful for fast computation of 2D convolution and correlation.
- It is reversible, i.e, we can revert to the initial domain.

Q Calculate 4-point DFT for the sequence $x(n) = \{0, 1, 2, 3\}$ using matrix method.

Ans. The 4-point DFT in one dimensional
= Kernel × Input sequence.

$$= \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -j & -1 & j \\ 1 & -1 & 1 & -1 \\ 1 & j & -1 & -j \end{bmatrix} \times \begin{bmatrix} 0 \\ 1 \\ 2 \\ 3 \end{bmatrix}$$

$$= \begin{bmatrix} 0+1+2+3 \\ 0-j-2+3j \\ 0-1+2-3 \\ 0+j-2-3j \end{bmatrix} = \begin{bmatrix} 6 \\ -2+2j \\ -2 \\ -2-2j \end{bmatrix}$$

Q3. Compute the 2D DFT of the grayscale image is given by

$$f(m,n) = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix}.$$

Ans. $F(k,l) = \text{Kernal} \times f(m,n) \times \text{Kernal}^T$

Kernal for the 4-point DFT is $= \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -j & -1 & j \\ 1 & -1 & 1 & -1 \\ 1 & j & -1 & -j \end{bmatrix}.$

$$F[k,l] = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -j & -1 & j \\ 1 & -1 & 1 & -1 \\ 1 & j & -1 & -j \end{bmatrix} \times \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix} \times \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -j & -1 & j \\ 1 & -1 & 1 & -1 \\ 1 & j & -1 & -j \end{bmatrix}.$$

$$F[k,l] = \begin{bmatrix} 4 & 4 & 4 & 4 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \times \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -j & -1 & j \\ 1 & -1 & 1 & -1 \\ 1 & j & -1 & -j \end{bmatrix}$$

$$= \begin{bmatrix} 16 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}.$$

DCT AND HAAR TRANSFORM

Discrete Cosine Transform

* One Dimensional DCT

$$x[k] = \alpha(k) \sum_{n=0}^{N-1} x(n) \cos\left\{\frac{(2n+1)\pi k}{2N}\right\}$$

where $0 \le k \le N-1$,

$$\alpha(k) = \begin{cases} \sqrt{\frac{1}{N}}, & \text{if } k=0 \\ \sqrt{\frac{2}{N}}, & \text{if } k \neq 0 \end{cases}$$

* Kernel of a 4-point DCT

$$\begin{bmatrix} 0.5 & 0.5 & 0.5 & 0.5 \\ 0.6532 & 0.2706 & -0.2706 & -0.6532 \\ 0.5 & -0.5 & -0.5 & 0.5 \\ 0.2706 & -0.6532 & 0.6533 & -0.2706 \end{bmatrix}$$

* 1D DCT : $F[k] = \text{kernel} \times f(x)$

* 2D DCT : $F[k,l] = \text{kernel} \times f(x,y) \times \text{kernel}^T$

Q Find the DCT of $f(x) = (1, 2, 4, 7)$.

Ans. F = Basis function $\times f(x)$

$$F = \begin{bmatrix} 0.5 & 0.5 & 0.5 & 0.5 \\ 0.6532 & 0.2706 & -0.2706 & -0.6532 \\ 0.5 & -0.5 & -0.5 & 0.5 \\ 0.2706 & -0.6533 & 0.6533 & -0.2706 \end{bmatrix} \times \begin{bmatrix} 1 \\ 2 \\ 4 \\ 7 \end{bmatrix}$$

$$= \begin{bmatrix} 7 \\ -4.459 \\ 1 \\ -0.370 \end{bmatrix}.$$

Q Find 2D DCT of $f(x, y) = \begin{bmatrix} 1 & 2 & 2 & 1 \\ 2 & 1 & 2 & 1 \\ 1 & 2 & 2 & 1 \\ 2 & 1 & 2 & 1 \end{bmatrix}$.

(2)

Ans: $F = \text{Kernel} \times f(x,y) \times \text{Kernel}^T$

$$F = \begin{bmatrix} 0.5 & 0.5 & 0.5 & 0.5 \\ 0.6532 & 0.2706 & -0.2706 & -0.6532 \\ 0.5 & -0.5 & -0.5 & 0.5 \\ 0.2706 & -0.6533 & 0.6533 & -0.2706 \end{bmatrix} \times \begin{bmatrix} 1 & 2 & 2 & 1 \\ 2 & 1 & 2 & 1 \\ 1 & 2 & 2 & 1 \\ 2 & 1 & 2 & 1 \end{bmatrix} \times$$

$$\begin{bmatrix} 0.5 & 0.6532 & 0.5 & 0.2706 \\ 0.5 & 0.2706 & -0.5 & -0.6533 \\ 0.5 & -0.2706 & -0.5 & 0.6533 \\ 0.5 & -0.6532 & 0.5 & -0.2706 \end{bmatrix}$$

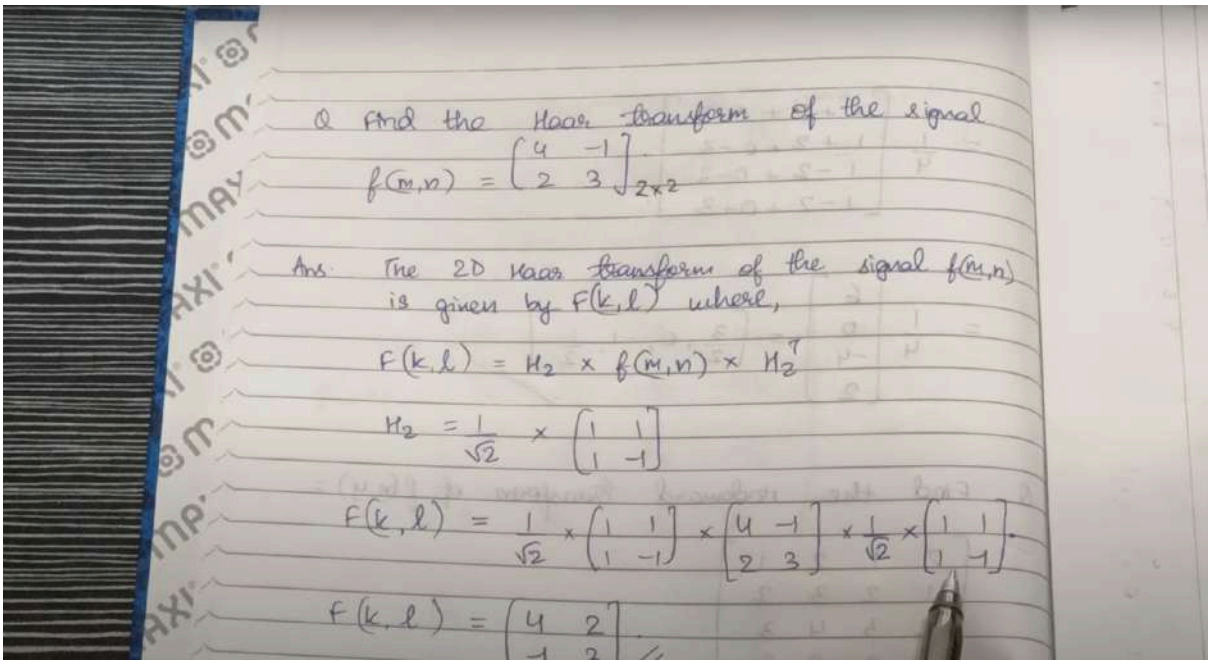
$$= \begin{bmatrix} 6 & 0.3025 & -1 & 0.9235 \\ 0 & -0.1463 & -0.3825 & -0.3532 \\ 0 & 0 & 0 & 0 \\ 0 & -0.3532 & -0.923 & -0.8525 \end{bmatrix}$$

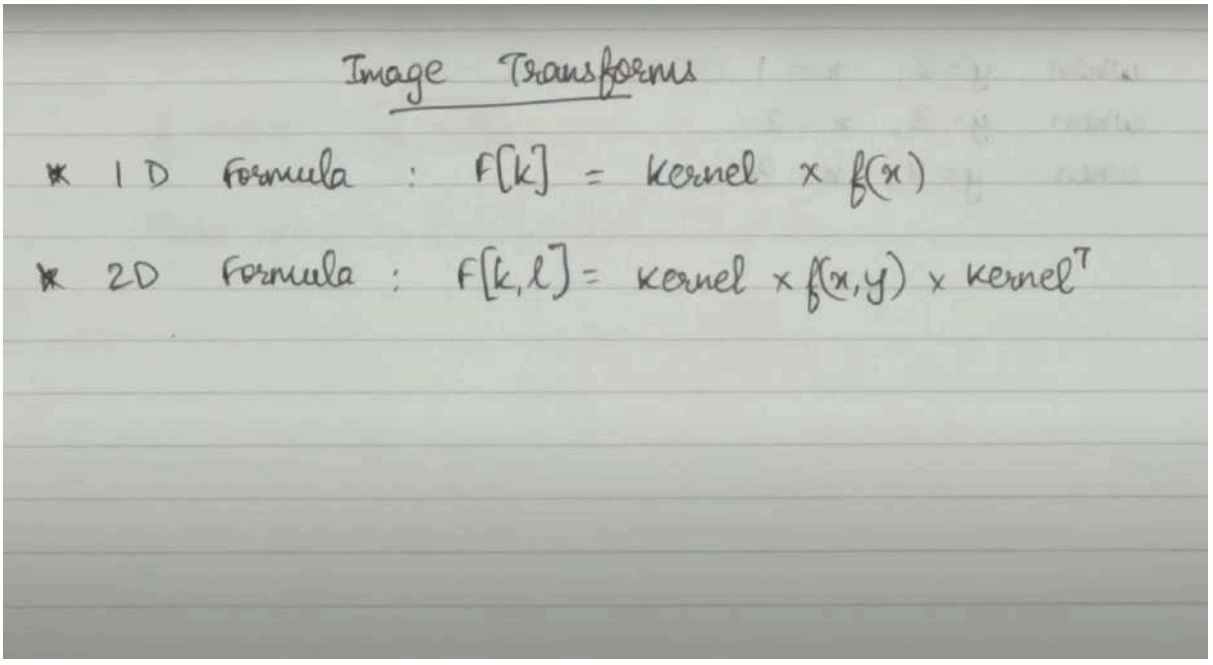
Haar Transform

* Kernel for a 2×2 matrix

$$H_2 = \frac{1}{\sqrt{2}} \times \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$

$$H_4 = \frac{1}{2} \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & -1 & -1 \\ \sqrt{2} & -\sqrt{2} & 0 & 0 \\ 0 & 0 & \sqrt{2} & -\sqrt{2} \end{bmatrix}$$





2.1 Basic Gray Level Transformations

2.1.1 Point operations:

These are transformations applied to individual pixels in an image, independent of their neighbors. The output value of a pixel depends only on its input value.

2.1.2 Contrast stretching:

This technique **expands** the **range of intensity levels in an image** to **improve contrast**. Expands the dynamic range of the image.

- Darkens dark pixels and brightens bright pixels.
- Can be achieved using linear or non-linear functions.

2.1.3 Clipping and thresholding:

Clipping **limits pixel values to a certain range**.

Limits the intensity values to a **specific range** by **setting values below** or **above** a **threshold** to **predefined** levels.

Thresholding **converts** a **grayscale image** to **binary** by **setting pixels above a threshold** to **one** value (e.g., **white**) and **below** to another (e.g., **black**)

2.1.4 Digital Negative:

- **Inverts** the **intensity values** of **pixels**.
- Converts dark pixels to bright pixels and vice versa.
- It's achieved by **subtracting each pixel value** from the **maximum intensity valu**e.

2.1.5 Intensity Level Slicing:

- This **highlights specific ranges** of **intensities** in an image while reducing or eliminating others.
- Can be **used** to **isolate objects** or **features** within an image.

2.1.6 Log Transformation:

- **Expands** the **darker pixel values** and **compresses** the **higher intensity values**, **making small details in dark areas more visible.**
- **Compresses the dynamic range of high-intensity values.**

s = log(r + 1)
where `s` is the output intensity and `r` is the input intensity.

2.1.7 Power law transformation:

Also known as **gamma correction**, this **transformation** is used to **adjust** the overall **brightness** of an image.
It's defined by the equation: s = c * r^γ, where r is the input intensity, s is the output, and γ **controls** the **shape** of the **curve**.
Adjusts the **pixel intensities based** on a **power law equation** to **enhance** or **reduce contrast**.

2.1.8 Bit Plane Slicing

- **Extracts individual bits** from the **binary representation** of **each pixel**.
- **Used** for **image analysis**, **compression**, and **watermarking**.
- Can be **used** to **visualize** the **contribution** of **different bits** to the **overall image appearance**.

2.1.9 Watermarking

Watermarking in digital image processing is the process of embedding additional information, often invisible to the human eye, into an image. This embedded information, known as a watermark, can be used for various purposes, including:

- Copyright protection: Identifying the owner of the image.
- Authentication: Verifying the image's authenticity and integrity.
- Tracking: Monitoring the distribution and usage of the image.
- Fingerprinting: Identifying specific copies of the image.

Types of Watermarks:

1. **Visible Watermarks:** Directly visible to the human eye, such as a logo or text superimposed on the image.

2. **Invisible Watermarks:** Embedded in the image data in a way that is not noticeable to the human eye, such as by modifying pixel values or adding noise.

Watermarking Techniques:

- **Spatial Domain Techniques:** Watermarks are embedded directly into the pixel values of the image.
- **Frequency Domain Techniques:** Watermarks are embedded into the frequency components of the image, often using transforms like Discrete Cosine Transform (DCT) or Discrete Wavelet Transform (DWT).
- **Spread Spectrum Techniques**: Watermarks are spread across the entire image, making them resistant to attacks.

Challenges in Watermarking:

- **Robustness**: The watermark should be resistant to various attacks, such as cropping, scaling, noise addition, and compression.
- **Invisibility**: The watermark should not be noticeable to the human eye or degrade the visual quality of the image.
- **Capacity**: The watermark should be able to carry sufficient information for its intended purpose.
- **Security**: The watermark should be difficult to remove or modify without detection.

Watermarking is a crucial technique in digital image processing, providing a means to protect intellectual property and track the usage of digital content.

2.2 Histogram Processing

A histogram is a **graphical representation** of the **distribution** of **pixel intensities** in an **image**. It **shows** how **many pixels** have a **particular intensity** value. Histogram processing **involves manipulating** the **histogram** to **enhance** or **modify** the **image's appearance**. It shows how many pixels in an image are **at each intensity level.**

2.2.1 Unnormalized Histogram:

This is the raw count of pixels at each intensity level. For an 8-bit grayscale image, it would have 256 bins (0-255), each containing the number of pixels with that intensity.

This is a plot that **shows** the **frequency** of **each intensity level (0 to 255 for an 8-bit image)** in the image. Each **value** on the **x-axis** represents an **intensity level**, while the **y-axis** represents the **number of pixels** with that **intensity**.

2.2.1 Normalized Histogram:

A normalized histogram in digital image processing is a type of histogram where the frequency of each pixel intensity is divided by the total number of pixels in the image. This ensures that the sum of all the values in the histogram equals 1, converting the frequency of pixel intensities into a probability distribution.

It represents the probability of a pixel having a particular intensity.

Normalization is useful for comparing images of different sizes.
This scales the histogram so that the sum of all frequencies equals 1.

If an image has 256 possible intensity levels (for an 8-bit grayscale image), the normalized histogram would show the probability of each intensity occurring, allowing for easy comparison between images of different sizes.

2.2.1 Histogram Equalization:

This is a technique used to **enhance image contrast** by **redistributing** the **intensity levels** more **evenly across** the **range**.
The process involves:
a) **Calculating** the cumulative distribution function (**CDF**) of the **normalized histogram**.
b) **Mapping** each **pixel intensity** to a **new value based** on the **CDF**.

The result is a more **balanced** and **uniformly distributed** histogram, making details in both **dark** and **bright** regions more visible.

2.2.2 Use of Histogram Statistics for Image Enhancement

- **Mean:** Represents the average intensity level. Can be used for global adjustments. Indicates average brightness.
- **Standard Deviation:** Measures the spread of intensity values. Can be used to control contrast. Represents contrast
- **Median:** Represents the middle intensity value. Can be used for noise reduction.
- **Mode:** Represents the most frequent intensity value. Can be used for highlighting dominant features.
- **Skewness**: Shows asymmetry in intensity distribution.
- **Kurtosis**: Indicates peakedness of the distribution.

2.3 Spatial operations

Spatial operations in image processing involve **manipulating pixel values based** on their **surrounding pixels** in the **image**.

This is typically done using a kernel (or mask) that moves across the image.

These operations are applied using filters, which are small matrices (or kernels) that "slide" over the image to compute a new pixel value.

These operations are fundamental to image processing and can be categorized into linear and nonlinear filters.

2.3.1 Basics of Spatial Filtering

- **Convolution:** The primary operation in spatial filtering. It involves sliding a kernel (filter mask) over the image and **computing** the **weighted sum** of **pixel values within** the **kernel's neighborhood**.
- **Kernel Size: Determines** the **extent** of the **neighborhood considered** in the **filtering process**. **Larger kernels result** in

more smoothing or **blurring**, while **smaller kernels preserve more details**.

2.3.2 Linear Filters

- **Spatial Low-Pass Smoothing Filters:**
    - **Low-pass filters:** smooth images by reducing high-frequency noise and details.
    - **Averaging:** Replaces each pixel with the average intensity of its neighbors. Reduces noise but also blurs edges.
    - **Weighted Averaging:** Assigns different weights to neighboring pixels based on their distance from the center pixel. Can be used to control the amount of smoothing.

2.3.3 Non-Linear Filters

These filters **apply non-linear operations** to the **neighborhood pixels**. Instead, they use other criteria such as **ranking** or **ordering** the pixel values.

a) **Median filter**: Replaces each pixel with the median value of its neighborhood. Effective for removing salt-and-pepper noise.

b) **Maximum and Minimum filters**: Replace each pixel with the maximum or minimum value in its neighborhood. Providing a balance between noise reduction and detail preservation.

c) **Midpoint filter**: Uses the average of the maximum and minimum values in the neighborhood.

d) **Alpha trimmed mean filter:** Calculates the mean after discarding the highest and lowest α values. Variants of the mean filter that exclude outliers to reduce the impact of noise.

2.3.4 High-Pass Sharpening Filters

- **High-pass filters** highlight fine details and edges by enhancing high-frequency components in the image.

- **High Boost Filter**: Enhances edges while preserving the overall appearance of the image by **amplifying** the **difference** between the **original** and **blurred images**.

  Amplifies the **high-frequency components** of the image to enhance edges and details.

- **High-Frequency Emphasis Filter**: A combination of **low-pass** and **high-pass** filtering that **enhances** high-frequency components but also preserves some low-frequency details for a balanced output.

  Combines a low-pass filter and a high-pass filter to **sharpen** the **image** while **preserving overall contrast**.

2.3.5 Gradient-based filters:

These filters are used for edge detection and image sharpening.
a) Robert Cross Gradient Operators: Simple 2x2 kernels for detecting edges.
b) Prewitt filters 3x3 kernels that compute gradients in horizontal and vertical directions.
c) Sobel filters: Similar to Prewitt, but with more weight on the central pixels.

2.3.6 Second Derivative Filters

- **Laplacian Filter:** A 3x3 kernel that approximates the second derivative of the image intensity. Can be used for edge detection and feature extraction. Highlight regions of rapid intensity change in all directions.

2.3.7 Magnification by replication and interpolation

Magnification by replication and **interpolation** are two methods used in image processing to increase the size (magnification) of an image.

Magnification in digital image processing involves increasing the size of an image without altering its content. This is achieved by adding new

pixels while preserving the original image information. Two common techniques for magnification are replication and interpolation.

1. Magnification by Replication

- Replication is the simplest method of magnifying an image. In this technique, each pixel is simply duplicated or "replicated" to fill the new image space.
- For example, if you want to double the size of an image, each pixel in the original image is copied into a 2x2 block of pixels in the larger image. This method results in a larger image but with noticeable blocky or "pixelated" edges, as no new pixel values are introduced.
- Pros: Simple and fast.
- Cons: Produces a blocky, pixelated appearance, especially noticeable at higher magnification levels.

2. Magnification by Interpolation

- Interpolation generates new pixel values based on surrounding pixel values when enlarging an image. This method results in smoother transitions between pixels compared to replication.
- Common interpolation techniques include:
    - **Nearest Neighbor Interpolation**: Similar to replication, it assigns the value of the nearest pixel to the new pixel position. However, unlike replication, it can handle more complex transformations.
    - **Bilinear Interpolation**: Averages the values of the four nearest surrounding pixels to compute a new pixel value. This method smooths out the image, reducing the blocky effect seen with replication.
    - **Bicubic Interpolation**: Uses a weighted average of the 16 nearest pixels, resulting in even smoother transitions and a higher quality enlarged image compared to bilinear interpolation.
- Pros: Produces smoother, higher-quality magnified images.
- Cons: More computationally intensive than replication, especially for bicubic interpolation.

Summary

- Replication simply duplicates pixels and is fast but results in blocky, low-quality magnification.
- Interpolation generates new pixels, providing a smoother and more natural-looking magnified image, though at the cost of increased computation.

Comparison:

Method	Pros	Cons
Replication	Simple and fast	Blocky appearance
Nearest Neighbor Interpolation	Simple	May introduce artifacts
Bilinear Interpolation	Smoother than nearest neighbor	Can introduce blurring
Bicubic Interpolation	Smoother and more accurate	Slower and more computationally expensive

3. Properties of Fourier Transform

3.1 Introduction to Fourier Transform and the Frequency Domain

Understanding Fourier Transform

Fourier Transform is a mathematical tool that decomposes a function into a sum of sine and cosine waves of different frequencies.

It helps us to analyze signals in the frequency domain, which can be very useful for understanding and processing various types of data.

It converts a signal from its original domain (usually time or space) to the frequency domain.

This allows us to analyze signals in terms of their frequency components rather than their time/space components.

A mathematical tool that transforms a time-domain or spatial-domain signal into its frequency-domain representation.

3.2 Frequency Domain

The analysis and representation of functions or signals based on frequency rather than time or space.

The frequency domain is a representation of a signal where the x-axis represents frequency and the y-axis represents the amplitude of each frequency component.

Applications: Fourier Transforms are widely used in signal processing, image analysis, audio compression, and more.

3.3. 1-D Continuous Fourier Transform

- **Definition:** The continuous Fourier Transform converts a continuous-time signal into a continuous spectrum of frequencies.

$$\int_{-\infty}^{\infty} f(t)\, e^{-jwt}\, dt$$

where:

- F(ω) is the Fourier Transform of f(t)
- ω is the angular frequency
- j is the imaginary unit (√-1)

Inverse Fourier Transform: Retrieves the original signal from its frequency-domain representation.

3.4. 2-D Continuous Fourier Transform

- **Application to Images:** A 2-D Fourier transform is used to analyze image data, converting spatial information into frequency information.
- **Formula:**

$$\int \int f(x,y)\, e^{-j(ux + vy)} dx.dy$$

- **Use in Image Processing:** Filters out noise, enhances image features, etc.

where:

- F(u, v) is the 2-D Fourier Transform of f(x, y)
- u and v are the spatial frequencies

The Fourier Transform and its inverse can be used to convert between the time/space domain and the frequency domain. This is a **fundamental concept** in **signal processing** and **image analysis**.

3.5 Fast Fourier Transform (FFT) and 2D Discrete Fourier Transform (DFT)

What is FFT?

- The Fast Fourier Transform (FFT) is an efficient algorithm for computing the Discrete Fourier Transform (DFT) and its inverse. The DFT transforms a sequence of complex numbers into another

sequence of complex numbers, allowing us to analyze the frequency components of discrete signals.
- **FFT** is a fast algorithm to compute the **Discrete Fourier Transform (DFT)** of a sequence, turning a signal from its original domain (often time or space) into a frequency domain.

3.6 FFT (Fast Fourier Transform):

- FFT reduces the time complexity of computing the DFT from **$O(N^2)$** to **$O(N \log N)$** for a 1D signal and from **$O(N^4)$** to **$O(N^2 \log N)$** for a 2D signal.
- It's much faster and is essential in applications like image processing, sound analysis, and compression.

3.6.1. 2D DFT and Visualization:

- The **2D DFT** is the Fourier transform applied to 2D data, like images.
- In 2D DFT, both rows and columns of an image are transformed. This helps convert an image from its pixel form into frequency components, allowing you to analyze patterns like edges or textures.

3.6.2. Time Complexity of DFT:

- Direct computation of DFT has a time complexity of **$O(N^2)$** for a 1D signal of length **N**.
- For a 2D signal (like an image of size NxN), DFT has a time complexity of **$O(N^4)$**, which can be slow for large data sets.

3.6.3. Derivation of 1D Fast Fourier Transform

- The **1D FFT** is based on the principle of **divide and conquer**:
    - The DFT can be split into two smaller DFTs: one for even-indexed elements and one for odd-indexed elements of the input sequence.

    - The FFT recursively applies this splitting until the problem size is reduced to 1.

Key steps:

1. Split the input sequence into even and odd parts.
2. Recursively apply FFT to both parts.
3. Combine the results using the twiddle factors (complex exponentials).

This results in a much faster algorithm than directly computing the DFT.

3.6.4. Time Complexity of FFT

- For a signal of length **N**, the FFT has a time complexity of **O(N log N)**.
- This efficiency comes from breaking down the DFT into smaller pieces (as mentioned earlier) and combining them in fewer steps.

3.6.5. Concept of Convolution

- **Convolution** is an operation that **combines** two signals to produce a third signal, showing how one signal modifies another.
- In simple terms, think of it as "sliding" one function over another and seeing how much they overlap at each point.

In discrete form, convolution sums the products of shifted versions of two sequences.

Example:

- In image processing, **convolution** is used for tasks like **blurring**, **sharpening**, and **edge detection**. For instance, when you apply a blur filter to an image, you’re convolving the image with a blur kernel (matrix).

3.6.7. Concept of Correlation

- **Correlation measures** the **similarity** between **two signals** or **datasets**. It tells you how one signal changes in relation to another.
- While convolution modifies a signal, correlation simply compares how similar two signals are as one is shifted over the other.

Example:

- In image processing, cross-correlation can be used to find where a smaller template (like a part of an image) matches inside a larger image.

3.6.8. Padding

- **Padding** involves adding extra elements (usually zeros) to a signal or image to modify its size for processing.
- It is commonly used when applying convolution or FFT, especially to ensure that the dimensions of the input match a required size or to prevent artifacts (like border effects).

Example:

- In 2D convolution (image filtering), padding is added to keep the output image the same size as the input image. Without padding, the image size would shrink after applying the filter.

Summary Table:

Topic	Definition	Time Complexity	Example
FFT (1D)	Fast algorithm to compute DFT	**O(N log N)**	Compressing an audio signal

2D DFT	Fourier transform for 2D data (e.g., images)	**O(N² log N)**	Image filtering, frequency analysis
Convolution	Combines two signals to produce a third signal	Depends on implementation	Image blurring, edge detection
Correlation	Measures similarity between two signals	Depends on size of data	Pattern recognition in images
Padding	Adds extra elements (zeros) to match size requirements	**O(N)** or less for each dimension	Ensures output size matches input size in convolution

3.7. 1D Discrete Fourier Transform (DFT)

Definition

The 1D DFT transforms a sequence of complex numbers into another sequence of complex numbers, representing the frequency components of the original sequence.

For a sequence x[n] of length N, the DFT is defined as:

$$X[k] = \sum_{n=0}^{N-1} x[n]\, e^{-j\frac{2\pi}{N} km}$$ where k = 0,1,2,3 ……

where:

- X[k] is the DFT output (frequency domain representation).
- x[n] is the input sequence (time domain representation).

- j is the imaginary unit.

Properties

1. **Linearity**: The DFT is a linear transformation.
2. **Periodicity**: The DFT output is periodic with a period of N.
3. **Symmetry**: For real-valued input signals, the DFT exhibits **conjugate symmetry**.

3.8. 2D Discrete Fourier Transform (2D DFT)

Definition

The 2D DFT extends the concept of the DFT to **two-dimensional signals,** such as images. It transforms a 2D array of complex numbers into another 2D array representing the frequency components.

For a 2D signal f(x,y) defined over a grid of size M×N, the 2D DFT is defined as:

$$F(u,v) = \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f(x, y)\, e^{-j\frac{2\pi}{M}ux}\, e^{-j\frac{2\pi}{N}vy}$$

where:

- F(u,v) is the 2D DFT output.
- f(x,y) is the input 2D signal.

Properties

1. **Linearity:** Similar to the 1D case, the 2D DFT is linear.
2. **Separability:** The 2D DFT can be computed as two successive 1D DFTs: first along the rows and then along the columns.
3. **Periodic:** The result is also periodic in both dimensions.

Applications

- Image processing (filtering, image reconstruction)

- Video processing
- Scientific computing (solving PDEs)

3.8 Other Transforms

The **Hadamard Transform**, **Haar Transform**, and **Discrete Cosine Transform (DCT)** are important tools in signal and image processing, each with its distinct use cases and properties:

1. **Hadamard Transform**:
    - A type of **orthogonal**, **non-sinusoidal** transform that uses **square waveforms instead** of **sine waves**.
    - The Hadamard matrix is used for performing the transformation, and the output represents the signal's decomposition into these square waves.
    - It's **widely used** in **error detection/correction**, **data compression**, and image processing due to its simplicity in computation.
2. **Haar Transform**:
    - A wavelet-based transform that **decomposes** a **signal** into **wavelets**, emphasizing **changes** in the signal **over time** or **space**.
    - The Haar wavelet is a **simple**, **piecewise constant function**, making this transform efficient for **hierarchical data analysis**, image compression, and denoising.
    - It’s often used in **multi-resolution analysis,** especially in the early stages of JPEG compression.
3. **Discrete Cosine Transform (DCT)**:
    - DCT **expresses** a **signal** in **terms** of **cosine functions oscillating** at **different frequencies**.
    - Most commonly used in image compression (e.g., JPEG), where it helps to separate the image into parts of differing importance.
    - The DCT is preferred for **compressing** data because it **concentrates** most of the **signal's information** in a **few coefficients**, making it effective for **lossy compression**.

4. Image Restoration and Compression

4.1 Image Restoration

Introduction: Image restoration is the process of **recovering** an **original** image from a **degraded version**. The goal is to **remove** or **reduce distortions** and **improve** the **quality** of the image.

4.2 Models for Image Degradation and Restoration Process:

1. **Degradation Model**: This represents how an original image I(x,y)is transformed into a degraded image D(x,y) through the addition of noise and other distortions. It can be expressed as:
   $D(x,y) = I(x,y) * H(x,y) + N(x,y)$
   where H(x,y)is the degradation function (often a blur), and N(x,y) is the noise.
2. **Restoration Process**: Involves applying techniques to reverse the degradation process. Common methods include:
   - **Inverse Filtering**: Attempts to reverse the effects of the degradation function.
   - **Wiener Filtering**: A statistical approach that minimizes the mean square error between the estimated and true images.

4.3 Noise Models

Noise is an unwanted variation in brightness or color in images. Various noise models include:

1. **Gaussian Noise**:
   - **Definition**: Characterized by a bell-shaped curve; it adds noise that follows a Gaussian distribution.
   - **Effect**: Causes random variations in pixel values, often seen in images captured in low-light conditions.

2. **Rayleigh Noise**:
    - **Definition**: Common in radar and sonar applications, where the amplitude of the noise follows a Rayleigh distribution.
    - **Effect**: Generally appears in images with high-frequency signals.
3. **Erlang Noise**:
    - **Definition**: A special case of the gamma distribution, often used in telecommunications.
    - **Effect**: Can model variations in signal strength.
4. **Exponential Noise**:
    - **Definition**: Characterized by an exponential distribution; often arises in situations like radioactive decay.
    - **Effect**: Common in images affected by high-frequency background noise.
5. **Uniform Noise**:
    - **Definition**: Each pixel value is equally likely within a certain range.
    - **Effect**: Introduces a consistent noise level across the image, making it look grainy.
6. **Impulse Noise**:
    - **Definition**: Also known as "salt-and-pepper" noise, it presents as random bright or dark pixels.
    - **Effect**: Can significantly distort images, making it difficult to discern details.

4.3 Estimation of Noise Parameters

Estimating noise parameters involves **analyzing** the **degraded image** to **quantify** the **noise present**. Common techniques include:

- **Statistical Methods**: Analyzing pixel distributions to estimate noise characteristics (e.g., variance for Gaussian noise).

- **Least Squares Estimation**: Using optimization techniques to minimize the difference between observed and expected pixel values.
- **Image Filtering Techniques**: Applying filters (like median filters) to separate noise from actual image content, allowing for better parameter estimation.

4.4 Restoration Filters

Restoration filters are used to **reduce noise** or **degradation** in **images**, aiming to **recover** a **cleaner version** of the **image**. These filters vary based on their **mathematical approach** and are **classified** into **different types**, **depending** on their **specific function**.

4.4.1. Mean Filters

Mean filters are simple noise reduction techniques that smooth the image by averaging pixel values in a neighborhood.

1. Arithmetic Mean Filter

- **Definition**: This filter computes the average of the pixels in the neighborhood of a central pixel.
- **Concept**: It smooths the image by replacing each pixel with the arithmetic mean of its surrounding pixels.
- **Example**: Used to reduce Gaussian noise, though it can blur edges. f(x,y)=n1i=1∑ngi
- where gi are the pixel values in the neighborhood.

4.4.2. Geometric Mean Filter

- **Definition**: The geometric mean filter replaces the pixel with the geometric mean of its surrounding pixels.
- **Concept**: Provides a more accurate representation of noise-free regions, especially for multiplicative noise.
- **Example**: Useful for reducing Gaussian or uniform noise.

- f(x,y)=(i=1∏ngi)1/n

4.4.3. Harmonic Mean Filter

- **Definition**: This filter replaces each pixel with the harmonic mean of its surrounding pixel values.
- **Concept**: Particularly effective in reducing salt noise (bright spots).
- **Example**: Effective for images with bright pixels or where noise is additive.
- f(x,y)=∑i=1ngi1n

4.4.4. Contraharmonic Mean Filter

- **Definition**: This filter can reduce both salt and pepper noise depending on the value of the order Q.
- **Concept**: Depending on the parameter Q, it reduces salt noise (for Q>0) or pepper noise (forQ<0).
- **Example**: Useful for impulse noise (salt-and-pepper).
- f(x,y)=∑giQ∑giQ+1

4.4.5. Order Statistics Filters

These filters operate on the sorted values of pixels in the neighborhood, useful for noise reduction and edge preservation.

1. Median Filter

- **Definition**: Replaces each pixel with the median of its surrounding pixels.
- **Concept**: Reduces impulse (salt-and-pepper) noise without significantly blurring the edges.
- **Example**: Highly effective in removing salt-and-pepper noise from images.

2. Min Filter

- **Definition**: Replaces each pixel with the minimum value in its neighborhood.
- **Concept**: Reduces pepper noise (black spots).

- **Example**: Useful for bright spot noise removal.

3. Max Filter

- **Definition**: Replaces each pixel with the maximum value in its neighborhood.
- **Concept**: Reduces salt noise (bright spots).
- **Example**: Effective for bright spot noise removal.

4. Midpoint Filter

- **Definition**: Replaces each pixel with the midpoint of the maximum and minimum pixel values in the neighborhood.
- **Concept**: Combines the effects of both Min and Max filters to reduce noise.
- **Example**: Effective for uniform noise.

5. Alpha Trimmed Mean Filter

- **Definition**: Discards a fixed number of the highest and lowest pixel values and averages the remaining ones.
- **Concept**: It reduces both Gaussian noise and impulse noise (by trimming extreme values).
- **Example**: Useful for mixed noise, balancing edge preservation with noise reduction.

4.5 Band Pass and Band Reject Filters

These filters are used in the **frequency domain** to **isolate** or **suppress certain frequency components** of an **image**.

1. Ideal Band Pass and Band Reject Filters

- **Definition**: Ideal filters that completely pass or reject frequencies in a certain range.
- **Concept**:
    - **Band Pass**: Passes only frequencies within a certain range.
    - **Band Reject**: Rejects frequencies within a certain range.

- **Example**: Used to remove specific frequencies like noise or isolate certain image details.

2. Butterworth Band Pass and Band Reject Filters

- **Definition**: Smoothly passes or rejects a range of frequencies with a gradual transition, unlike ideal filters.
- **Concept**: Provides a smoother frequency transition, avoiding the sharp cutoff of ideal filters.
- **Example**: Applied to reduce noise in images while preserving details without introducing ringing effects.

3. Gaussian Band Pass and Band Reject Filters

- **Definition**: Gaussian filters pass or reject frequencies using a Gaussian distribution curve.
- **Concept**: Provides the smoothest transition between frequencies, minimizing distortions.
- **Example**: Applied when smooth, non-distorted filtering is required, such as in medical imaging or satellite imagery.

Summary of Usage

- **Ideal filters**: Provide sharp transitions but can introduce artifacts like ringing.
- **Butterworth filters**: Offer a balance between smoothness and sharpness, reducing artifacts while still controlling frequency transitions.
- **Gaussian filters** provide the smoothest transitions and are ideal for minimizing noise and distortions, especially in sensitive applications like medical or satellite imagings.

Unit 5: Morphological Operations

Introduction to Morphological Image Processing

Definition: Morphological image processing is a **technique** that **analyzes** and **processes geometric structures** within an image. It

focuses on the **shape** or **structure** of **objects rather** than their **color** or **intensity**.

Concept: Morphological operations use a **structuring element (a small shape or kernel)** to **probe** and **transform** the input image. The two primary operations are **hit** and **fit**:

- **Hit**: Refers to the operation where the structuring element fits within the shape in the image. It emphasizes the parts of the image where the structuring element can completely overlap with the object.
- **Fit**: Involves how well the structuring element matches the shape in the image, often focusing on the presence of shapes.

Example: In a binary image of the letter "A," a structuring element shaped like a small rectangle can be used to determine where the letter fits and where it can hit, helping to identify its outline and structure.

Dilation and Erosion

Dilation: Definition: Dilation is a morphological operation that adds pixels to the boundaries of objects in a binary image.

Concept: It expands the shape of an object, making it larger. The operation increases the area of foreground pixels based on the structuring element.

Example: In a binary image of a small circle, applying dilation with a circular structuring element will make the circle larger, potentially filling in small gaps.

Erosion: Definition: Erosion is the opposite of dilation; it removes pixels from the boundaries of objects.

Concept: This operation shrinks the shape of an object by eroding away pixels from the edges, effectively reducing the area of foreground pixels.

Example: In the same binary image of a small circle, applying erosion will make the circle smaller, potentially removing noise and thin protrusions.

Opening and Closing

Opening: Definition: Opening is a morphological operation that **consists** of **erosion followed** by **dilation**.

Concept: It removes small objects (noise) from an image while preserving the shape and size of larger objects. Opening smooths the contours of objects.

Example: In a binary image with small noise points around a large object, applying an opening will remove the noise while keeping the larger object intact.

Closing: Definition: Closing is the reverse of opening; it consists of **dilation** followed by **erosion**.

Concept: This operation fills small holes and gaps in an object while preserving the overall shape. It helps connect nearby objects or smooth contours.

Text Enhancement:

- **Image:** A blurry image with text.
- **Process:** Dilate the image to thicken the text strokes, then erode it to remove excess pixels.
- **Result:** The text becomes more legible

Unit 6: Image Segmentation (8 Hrs.)

1. Point Detection

Definition: Point detection involves identifying isolated pixels with an intensity value significantly different from their neighboring pixels. It is used to find specific features like corners or point-like structures.

- **Concept:** Detecting sharp changes in intensity around a single pixel.
- **Example:** In a medical image, point detection can identify small bright spots (e.g., micro-calcifications in mammograms).

2. Line Detection

Definition: Line detection is used to **find linear features** or edges by identifying changes in **intensity** along a **specific direction.**

- **Concept:** Identifies **continuous changes** in **pixel values along a line**.
- **Example:** Detecting roads or rivers in satellite imagery, where lines represent significant features.

3. Edge Detection using Gradient Filters

Definition: Edge detection identifies regions in the image with a sharp change in intensity, representing object boundaries.

- **Concept:** The gradient filter calculates the rate of change in pixel intensity (first derivative).
- **Example:** Sobel or Prewitt operators are common gradient-based filters used to detect edges in images. For instance, detecting the boundary of an object like a car in a traffic image.

4. Edge Detection using Laplacian Filters

Definition: The Laplacian filter detects edges by finding regions where the intensity changes most rapidly, i.e., by identifying second-order derivatives (rate of change of the gradient).

- **Concept:** A zero-crossing occurs when intensity values change rapidly.
- **Example:** In image sharpening, Laplacian filters highlight edges by detecting points of sudden change in intensity, helping clarify object boundaries.

5. Mexican Hat Filters

Definition: Also known as the *Laplacian of Gaussian (LoG)*, the Mexican Hat filter smooths the image with a Gaussian and then applies a Laplacian filter to detect edges.

- **Concept:** Combines smoothing and edge detection in a single step.
- **Example:** It is used in blob detection, such as identifying small round objects in astronomy images (stars, planets).

6. Edge Linking and Boundary Detection

Definition: After detecting edges, edge linking ensures that edges form continuous and closed boundaries, representing the shape of objects in the image.

- **Concept:** Uses algorithms like thresholding and neighborhood analysis to link edge points.

Definition: Edge linking is the process of connecting detected edges in an image to **form continuous boundaries**, while boundary detection involves **identifying** the **outlines** that **separate distinct regions** within an image.

Main Concept:

- Edge Linking: After detecting edges (sharp changes in intensity), this technique connects these edges to create complete shapes or contours of objects.
- Boundary Detection: Focuses on locating the edges that define the boundaries between different regions or objects in an image.

Example: In a photo of a tree:

- Edge Linking: Detects the edges of the tree's trunk and branches and connects them to form a continuous outline.
- Boundary Detection: Identifies the boundary between the tree and the background (e.g., sky and grass).

7. Hough Transform

Definition: Hough Transform is a technique to **detect shapes like lines, circles, or ellipses** by **converting image points into a parameter space.**

- **Concept:** Detects shapes by mapping image pixels to a mathematical representation (e.g., points on a line to the slope-intercept form).
- **Example:** It is commonly used to detect lanes on roads in autonomous driving systems, where the lines form lane boundaries.

These concepts are fundamental to edge-based image segmentation, where the goal is to partition an image based on changes in intensity values.

Thresholding

Definition: Thresholding is a technique used to separate foreground objects from the background by converting a grayscale image into a binary image based on pixel intensity values.

Types of Thresholding:

- **Global Thresholding:**
  - **Definition:** A single threshold value is applied to the entire image. Pixels above the threshold are classified as one category (e.g., foreground), and those below as another (e.g., background).
  - **Concept:** Simple but effective for images with clear intensity differences.
  - **Example:** In a document scanning process, global thresholding can separate text from the background by setting a fixed intensity threshold for the entire image.

- **Local (or Regional) Thresholding:**
  - **Definition:** Different threshold values are used for different regions of the image, which allows adaptation to varying lighting or intensity changes.

    - **Concept:** Useful for images where illumination is not uniform.
    - **Example:** In an image with shadows, local thresholding can segment regions more accurately by adjusting thresholds for each region individually, ensuring even shadowed parts are processed correctly.
- **Adaptive Thresholding:**
    - **Definition:** The threshold value is dynamically determined for small regions of the image based on local pixel intensities.
    - **Concept:** It adapts to changes in lighting conditions, automatically setting thresholds for different regions.
    - **Example:** In a natural scene with uneven lighting (e.g., an outdoor image with both sunlit and shaded areas), adaptive thresholding can detect objects across the image without being affected by lighting variations.

2. Region-Based Segmentation

Definition: Region-based segmentation techniques group pixels into larger regions based on predefined criteria, such as intensity or color similarity.

Types of Region-Based Segmentation:

- **Region Growing Algorithm:**
    - **Definition:** This technique starts with a seed pixel and grows a region by adding neighboring pixels that are similar to the seed based on some criterion (like intensity, color, or texture).
    - **Concept:** Iteratively includes neighboring pixels if they are similar to the seed, forming a coherent region.
    - **Example:** In medical imaging, region growing can be used to segment tumors or organs by starting from a pixel inside the tumor/organ and growing until the boundary is reached.
- **Region Split and Merge Algorithm:**
    - **Definition:** The image is initially divided into small regions (splitting), and then neighboring regions are merged if they are similar based on certain criteria (like intensity or texture).

- **Concept:** Divides regions that are too heterogeneous and merges those that are homogeneous.
- **Example:** In satellite imagery, where different land areas may be heterogeneous, this algorithm can help distinguish between distinct regions (e.g., water, forest, urban areas) by splitting and merging based on similarity.

Process of Region Split and Merge:

- **Split:** If a region is too varied (heterogeneous), it is recursively split into smaller subregions.
- **Merge:** Once the regions are small enough, they are merged with adjacent regions if they share similar properties.

In **similarity-based segmentation**, the focus is on grouping pixels that share similar properties. Thresholding works best when there's a clear distinction in intensity, while region-based segmentation is ideal for grouping areas based on pixel similarity criteria.

1. Representation

Definition: Representation refers to the process of converting segmented regions (e.g., objects) into a form that is easier to analyze and describe. It is how the shape or boundary of an object is stored or modeled.

- **Concept:** After segmenting an object, the next step is to represent its shape or boundary efficiently for further analysis or recognition.
- **Example:** A binary image can be represented as a series of coordinates describing its boundary or as a mathematical model.

2. Description

Definition: Description involves extracting meaningful features from the representation to characterize the shape, size, texture, or other attributes of an object.

- **Concept:** Descriptors are used to describe an object’s shape or boundary compactly, allowing comparison and recognition.

- **Example:** A shape's boundary might be described using its length, curvature, or specific points of interest (e.g., corners).

3. Recognition

Definition: Recognition is the process of classifying an object based on its description. The extracted features are used to compare objects and identify them.

- **Concept:** Once an object is described, the description is compared with known objects in a database to recognize or classify the object.
- **Example:** In facial recognition, descriptors derived from facial features (e.g., distances between key points) are compared to a database of known faces.

4. Introduction to Some Descriptors:

a. Chain Codes

- **Definition:** Chain codes describe the boundary of an object by encoding the direction of movement along the boundary, typically using a series of symbols that correspond to directions in a grid (e.g., 0 for right, 1 for up).
- **Concept:** The object's boundary is traced, and the direction of movement between successive boundary points is encoded as a sequence.
- **Example:** A square object in an image can have its boundary described using the directions of movement between pixels: right, up, left, and down. This is represented by a chain of codes like [0, 1, 2, 3].

b. Signatures

- **Definition:** A signature is a one-dimensional representation of a shape, typically obtained by plotting a feature like the distance from the center of an object to its boundary as a function of angle.
- **Concept:** It converts a 2D shape into a 1D signal by using features like the radius from the centroid to the boundary at different angles.

- **Example:** For a circular object, the signature would be a constant value representing the radius. For more complex shapes, the signature changes with the angle.

c. Shape Numbers

- **Definition:** Shape numbers describe the geometry of an object's boundary by representing the shape as a series of digits or codes, often derived from chain codes.
- **Concept:** By encoding the sequence of boundary directions as numbers, the object's shape can be uniquely identified.
- **Example:** For a rectangular object, shape numbers represent the sequence of turns needed to trace the boundary, which can be used to compare with other shapes.

d. Fourier Descriptors

- **Definition:** Fourier descriptors are used to describe an object's shape by **transforming** the **boundary** into the **frequency domain using** the **Fourier Transform**. The resulting coefficients capture the shape's essential features.
- **Concept:** It **represents** the **shape** as a **series** of **sinusoidal components** (**frequencies**), **allowing** for **efficient comparison** and **shape matching**.
- **Example:** Two objects that are similar in shape but rotated or scaled can have similar Fourier descriptors, making this technique useful for recognizing shapes regardless of orientation or size.

Summary

- **Representation** converts a segmented region into a form suitable for further processing (e.g., boundary coordinates).
- **Description** involves extracting features from the representation to characterize the object (e.g., boundary length, curvature).
- **Recognition** uses the description to classify or identify the object.

The **descriptors** (Chain Codes, Signatures, Shape Numbers, Fourier Descriptors) provide various ways to describe and analyze the shape and boundary of objects for tasks like object recognition and comparison.

Simplification

Chain Codes

- **What It Is:** Chain codes describe the outline (boundary) of an object by encoding the direction in which you move from one boundary point to the next.
- **How It Works:** Imagine tracing the edge of a shape. As you move along the edge, you record the direction you're moving (e.g., right, up, left). These directions are turned into a sequence of numbers.
- **Example:** If you trace a square, you might move right (0), up (1), left (2), and down (3), so the chain code for the square would be something like [0, 1, 2, 3].

b. Signatures

- **What It Is:** A signature simplifies a shape by looking at how far the edge is from the center of the object at different angles.
- **How It Works:** You start at the center of the shape and measure the distance to the boundary at different points (like spokes on a wheel). These distances create a "signature" or a graph that represents the shape.
- **Example:** If the object is a perfect circle, the distance from the center to the edge is always the same, so the signature would be a flat line. If the object is a star, the distance will change, making the signature go up and down.

c. Shape Numbers

- **What It Is:** Shape numbers are a way of summarizing the shape of an object using a sequence of numbers that represent the angles or turns made as you trace its outline.
- **How It Works:** As you trace the boundary of an object, you count the number of "turns" or changes in direction, and those turns are converted into numbers. This number sequence is like a "fingerprint" for the shape.

- **Example:** If you trace a rectangle, your shape number might describe the sharp turns at each corner. You can then compare these numbers to other objects to see if they have similar shapes.

d. Fourier Descriptors

- **What It Is:** Fourier descriptors transform the shape of an object into a bunch of waves (like sound waves) to capture the important details of its shape.
- **How It Works:** By breaking down the shape's outline into different frequency components (waves), you can create a compact description of the shape. This helps in comparing shapes, even if they're rotated or resized.
- **Example:** If you have two shapes that are similar but one is rotated, their Fourier descriptors will still look similar, making it easy to recognize that they are the same shape.

Pattern Recognition

Overview: Pattern recognition is the **process** of **identifying patterns** and **regularities** in data. It **involves classifying input data** into **predefined categories** or **classes** based on the **features** extracted from the data.

Block Diagram:

1. **Input Data**: Raw data (e.g., images, signals)
2. **Preprocessing**: Noise reduction, normalization
3. **Feature Extraction**: Identifying and extracting relevant features
4. **Pattern Classification**: Using algorithms to classify the patterns
5. **Output**: Class labels or decisions

Patterns and Pattern Classes

Patterns: Patterns are **recognizable arrangements** or **configurations** in **data** (e.g., **shapes** in an **image**, **sounds** in **audio**).

Pattern Classes: These are **groups** or **categories** that **patterns** are **classified** into **based** on **similarities**. For instance, in **image recognition**, **classes** can be "**cats**," "**dogs**," "**cars**," etc.

Decision-Theoretic Methods

Definition: Decision-theoretic methods **utilize statistical principles** to make decisions based on **uncertain information**. It involves:

- **Modeling**: Constructing models for the likelihood of different patterns.
- **Decision Rules**: Establishing rules to minimize error or maximize accuracy.

Example: In face recognition, the method calculates probabilities of a face belonging to known classes and selects the class with the highest probability.

Introduction to Neural Networks

Definition: Neural networks are computational models inspired by the human brain, consisting of interconnected nodes (neurons) organized in layers (input, hidden, output).

Concept: Neural networks learn from data by adjusting weights based on error feedback. They can capture complex patterns and relationships.

Example: A simple neural network can classify handwritten digits by learning from labeled examples (0-9).

Neural Network-Based Image Recognition

Definition: This involves using neural networks to identify and classify objects in images.

Concept: Convolutional Neural Networks (CNNs) are commonly used for image recognition. They automatically learn spatial hierarchies of features through convolutional layers, pooling layers, and fully connected layers.

Example: A CNN trained on the MNIST dataset can accurately recognize handwritten digits by processing the image's pixels and identifying key features like edges and shapes.

Descriptive

Pattern Recognition

Overview: Pattern recognition is a branch of machine learning that focuses on classifying data based on its features. It encompasses various techniques to **identify patterns, trends, or regularities in data**, which can be applied to various domains such as image processing, speech recognition, and text analysis. The primary goal is to map input data to predefined categories or classes.

Block Diagram:

1. **Input Data**: This is the raw information collected, such as images, audio signals, or sensor readings.
2. **Preprocessing**: This stage involves cleaning and preparing the data. Techniques like noise reduction, normalization, and data augmentation are used to improve data quality.
3. **Feature Extraction**: In this step, relevant features are extracted from the preprocessed data. This could involve detecting edges, shapes, or textures in images, which help in distinguishing between different patterns.
4. **Pattern Classification**: Here, classification algorithms (like decision trees, SVMs, or neural networks) analyze the extracted features to classify the data into predefined categories.
5. **Output**: The final step produces the class labels or decisions based on the classification results, indicating which category the input data belongs to.

Patterns and Pattern Classes

Patterns: In the context of pattern recognition, a pattern refers to a specific arrangement of data that can be recognized and categorized. For example, in image processing, a pattern could be the specific arrangement of pixels that represent a particular shape or object.

Pattern Classes: These are the different categories into which patterns can be grouped. Each class represents a distinct group of similar patterns. For instance, in a dataset of animal images, you might have classes like "cats," "dogs," "birds," etc. The goal of a pattern recognition system is to accurately classify new input data into one of these classes based on the learned patterns.

Decision-Theoretic Methods

Definition: Decision-theoretic methods are frameworks that utilize statistical principles to make informed decisions in the presence of uncertainty. These methods are particularly useful in pattern recognition, where the data can be noisy or incomplete.

Modeling: The process begins with creating statistical models that represent the likelihood of different patterns occurring. This could involve estimating probability distributions for the classes based on training data.

Decision Rules: Once the models are established, decision rules are formulated to classify new data points. A common approach is to use the Maximum A Posteriori (MAP) criterion, which selects the class with the highest posterior probability given the observed data.

Example: In a face recognition system, the decision-theoretic approach would involve calculating the probabilities that a given image belongs to each known person and classifying it as the person with the highest probability.

Introduction to Neural Networks

Definition: Neural networks are computational models inspired by the structure and function of the human brain. They consist of interconnected groups of nodes (neurons) organized into layers: an input layer, one or more hidden layers, and an output layer.

Concept: Neural networks learn by adjusting the weights of connections between neurons based on the error of predictions during training. This is typically achieved through a process called backpropagation, where the network iteratively updates weights to minimize prediction errors.

Example: A simple neural network can be trained on a dataset of handwritten digits (like MNIST) to recognize and classify the digits from images. The network learns to identify patterns associated with each digit by adjusting its weights during training.

Neural Network-Based Image Recognition

Definition: This refers to the use of neural networks, particularly Convolutional Neural Networks (CNNs), for identifying and classifying objects within images. CNNs are specially designed to process and analyze visual data.

Concept: CNNs use layers that apply convolution operations to extract features from images. These layers can capture spatial hierarchies by detecting edges, textures, and higher-level features (like shapes and objects) through successive layers. Pooling layers reduce dimensionality and help in focusing on the most important features.

Example: A CNN trained on the CIFAR-10 dataset can recognize various objects, such as airplanes, cars, and animals. By processing an image of a cat, the CNN extracts features through its layers and eventually classifies the image as belonging to the "cat" category based on learned patterns.