Given the following 3x3 Gradient Magnitude and Gradient Direction matrices:
Gradient Magnitude Matrix (M):
50 70 30
40 100 20
60 90 50
Gradient Direction Matrix (θ):
45 90 135
0 45 90
135 0 45
Apply Non-Maximum Suppression to the given gradient magnitude and direction matrices.
Use the following Double Thresholding: High Threshold = 80 & Low Threshold = 30
Classify the pixels into:Strong edges, Weak edges, Non-edges.Provide the final edge classification for each pixel.

1. Non-Maximum Suppression (NMS) and Double Thresholding

The Non-Maximum Suppression (NMS) and Double Thresholding steps are part of the **Canny Edge Detection** algorithm. The goal is to thin the edges (NMS) and then classify them (Double Thresholding).

A. Non-Maximum Suppression (NMS)

In NMS, for each pixel in the **Gradient Magnitude Matrix (M)**, we check if its magnitude is a local maximum along the direction of the gradient (θ).

Rules for NMS:

1. **Quantize the Direction:** The gradient direction θ is typically quantized to one of four sectors: 0° (Horizontal), 45° (Diagonal A), 90° (Vertical), and 135° (Diagonal B).
   - 0° or 180°: Compare with **Left** and **Right** neighbors.
   - 45°: Compare with **Top-Left** and **Bottom-Right** neighbors.
   - 90° or 270°: Compare with **Top** and **Bottom** neighbors.
   - 135° or 315°: Compare with **Top-Right** and **Bottom-Left** neighbors.
2. **Comparison:** If the center pixel's magnitude is **not strictly greater** than **both** neighbors in the gradient direction, its magnitude is set to 0 (suppressed). Otherwise, it retains its original magnitude.

Since this is a 3×3 matrix, only the **center pixel** $M(1,1)$ can be fully processed, as the border pixels lack all necessary neighbors.

$$M = \begin{pmatrix} 50 & 70 & 30 \\ 40 & \mathbf{100} & 20 \\ 60 & 90 & 50 \end{pmatrix}, \quad \theta = \begin{pmatrix} 45 & 90 & 135 \\ 0 & \mathbf{45} & 90 \\ 135 & 0 & 45 \end{pmatrix}$$

Processing Center Pixel $M(1,1) = 100$:

- **Gradient Direction** $\theta(1,1) = 45°$.
- This direction points towards the **Top-Left** and **Bottom-Right** neighbors.
- **Neighbors:** $M(0,0) = 50$ (Top-Left) and $M(2,2) = 50$ (Bottom-Right).
- **Comparison:** $M(1,1) = 100$ is compared to $\{50, 50\}$.
- $100 > 50$ and $100 > 50$. The magnitude is a local maximum.
- **Result:** $M_{NMS}(1,1) = 100$.

For the **border pixels**, we cannot perform NMS accurately due to missing neighbors. For a typical exam scenario, border pixels either retain their value or NMS is simply not performed on them. Assuming **border pixels retain their original value for classification**:

$$\text{NMS Magnitude Matrix } (M_{NMS}) = \begin{pmatrix} 50 & 70 & 30 \\ 40 & \mathbf{100} & 20 \\ 60 & 90 & 50 \end{pmatrix}$$

B. Double Thresholding and Classification

- **High Threshold** $(T_H) = 80$
- **Low Threshold** $(T_L) = 30$

The classification of pixels is as follows:

- **Strong Edge:** If $M_{NMS} \geq T_H$
- **Weak Edge:** If $T_L \leq M_{NMS} < T_H$
- **Non-Edge:** If $M_{NMS} < T_L$

Magnitude M_{NMS}	Classification
50	**Weak** $(30 \leq 50 < 80)$
70	**Weak** $(30 \leq 70 < 80)$
30	**Weak** $(30 \leq 30 < 80)$
40	**Weak** $(30 \leq 40 < 80)$
100	**Strong** $(100 \geq 80)$
20	**Non-Edge** $(20 < 30)$
60	**Weak** $(30 \leq 60 < 80)$
90	**Strong** $(90 \geq 80)$

Given a 2D image represented as:
[5 2
3 8]
Calculate the 2D Discrete Fourier Transform (DFT) for the given image.

Given image $f(x, y)$:

$$f(x, y) = \begin{pmatrix} 5 & 2 \\ 3 & 8 \end{pmatrix}$$

The 2D DFT $F(u, v)$ for an $M \times N$ image is:

$$F(u, v) = \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f(x, y) e^{-j2\pi\left(\frac{ux}{M} + \frac{vy}{N}\right)}$$

Here, $M = 2$ and $N = 2$. The indices are $x, y, u, v \in \{0, 1\}$.

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

The four coefficients are $F(0, 0)$, $F(0, 1)$, $F(1, 0)$, and $F(1, 1)$.

A. $F(0, 0)$ (DC Component)

For $u = 0, v = 0$:

$$F(0, 0) = \sum_{x=0}^{1} \sum_{y=0}^{1} f(x, y) e^{0} = f(0, 0) + f(0, 1) + f(1, 0) + f(1, 1)$$

$$F(0, 0) = 5 + 2 + 3 + 8 = \mathbf{18}$$

B. $F(0, 1)$

For $u = 0, v = 1$:

$$F(0,1) = \sum_{x=0}^{1} \sum_{y=0}^{1} f(x,y) e^{-j\pi y}$$

- $y = 0$: $f(x,0)e^0 = f(0,0) + f(1,0) = 5 + 3 = 8$
- $y = 1$: $f(x,1)e^{-j\pi} = f(x,1)(-1) = (2+8)(-1) = -10$

$$F(0,1) = 8 + (-10) = -\mathbf{2}$$

C. $F(1, 0)$

For $u = 1, v = 0$:

$$F(1,0) = \sum_{x=0}^{1} \sum_{y=0}^{1} f(x,y) e^{-j\pi x}$$

- $x = 0$: $f(0,y)e^0 = f(0,0) + f(0,1) = 5 + 2 = 7$
- $x = 1$: $f(1,y)e^{-j\pi} = f(1,y)(-1) = (3+8)(-1) = -11$

$$F(1,0) = 7 + (-11) = -\mathbf{4}$$

D. $F(1,1)$

For $u=1, v=1$:

$$F(1,1)=\sum_{x=0}^{1}\sum_{y=0}^{1} f(x,y)e^{-j\pi(x+y)}$$

- $x=0, y=0$: $f(0,0)e^{0}=5$
- $x=0, y=1$: $f(0,1)e^{-j\pi}=2(-1)=-2$
- $x=1, y=0$: $f(1,0)e^{-j\pi}=3(-1)=-3$
- $x=1, y=1$: $f(1,1)e^{-j2\pi}=8(1)=8$

$$F(1,1)=5-2-3+8=\mathbf{8}$$

2D DFT Matrix $F(u,v)$:

$$F(u,v)=\begin{pmatrix} F(0,0) & F(1,0) \\ F(1,0) & F(1,1) \end{pmatrix}=\begin{pmatrix} 18 & -2 \\ -4 & 8 \end{pmatrix}$$

Last normalize with 1/MN:[18/4 -2/4]
[$-$ 4/4 8/4]

Apply a 3x3 Gaussian filter with the following kernel to an image patch:
[1 2 1
2 4 2
1 2 1]
to the image region:
[10 20 30
40 50 60
70 80 90]
Normalize the kernel by dividing by 16.

Step 1: Normalize the Kernel

The **Normalized Gaussian Kernel (K)** is $K = \frac{1}{16} K_{\text{unnormalized}}$.

$$K = \begin{pmatrix} 1/16 & 2/16 & 1/16 \\ 2/16 & 4/16 & 2/16 \\ 1/16 & 2/16 & 1/16 \end{pmatrix} = \begin{pmatrix} 0.0625 & 0.125 & 0.0625 \\ 0.125 & 0.25 & 0.125 \\ 0.0625 & 0.125 & 0.0625 \end{pmatrix}$$

Step 2: Apply Zero-Padding

To calculate a filtered value for all 3×3 pixels using a 3×3 kernel, the original image must be zero-padded by a border of 1 pixel on all sides. This creates a 5×5 padded image.

Step 3: Perform Convolution and Calculate Output Matrix

The output pixel $O(i, j)$ is the convolution result when the kernel is centered over the original pixel $I(i, j)$, calculated as the weighted sum of the 3×3 neighborhood covered by the kernel, divided by 16.

The calculation for each of the 9 pixels is detailed below (Weighted Sum divided by 16):

Row (i)	Col (j)	Original Pixel	Neighbors in Padded Image	Weighted Sum (Numerator)	Final Filtered Value
0	0	10	$\begin{matrix} 0 & 0 & 0 \\ 0 & \mathbf{10} & 20 \\ 0 & 40 & 50 \end{matrix}$	$(10 \cdot 4) + (20 \cdot 2) + (40 \cdot 2) + (50 \cdot 1) = 210$	$210/16 = \mathbf{13.125}$
0	1	20	$\begin{matrix} 0 & 0 & 0 \\ 10 & \mathbf{20} & 30 \\ 40 & 50 & 60 \end{matrix}$	$(10 \cdot 2) + (20 \cdot 4) + (30 \cdot 2) + (40 \cdot 1) + (50 \cdot 2) + (60 \cdot 1) = 360$	$360/16 = \mathbf{22.500}$
0	2	30	$\begin{matrix} 0 & 0 & 0 \\ 20 & \mathbf{30} & 0 \\ 50 & 60 & 0 \end{matrix}$	$(20 \cdot 2) + (30 \cdot 4) + (50 \cdot 1) + (60 \cdot 2) = 330$	$330/16 = \mathbf{20.625}$
1	0	40	$\begin{matrix} 0 & 10 & 20 \\ 0 & \mathbf{40} & 50 \\ 0 & 70 & 80 \end{matrix}$	$(10 \cdot 2) + (20 \cdot 1) + (40 \cdot 4) + (50 \cdot 2) + (70 \cdot 2) + (80 \cdot 1) = 520$	$520/16 = \mathbf{32.500}$
1	1	50	$\begin{matrix} 10 & 20 & 30 \\ 40 & \mathbf{50} & 60 \\ 70 & 80 & 90 \end{matrix}$	$(10 \cdot 1) + \cdots + (90 \cdot 1) = 800$	$800/16 = \mathbf{50.000}$
1	2	60	$\begin{matrix} 20 & 30 & 0 \\ 50 & \mathbf{60} & 0 \\ 80 & 90 & 0 \end{matrix}$	$(20 \cdot 1) + (30 \cdot 2) + (50 \cdot 2) + (60 \cdot 4) + (80 \cdot 1) + (90 \cdot 2) = 680$	$680/16 = \mathbf{42.500}$
2	0	70	$\begin{matrix} 0 & 40 & 50 \\ 0 & \mathbf{70} & 80 \\ 0 & 0 & 0 \end{matrix}$	$(40 \cdot 2) + (50 \cdot 1) + (70 \cdot 4) + (80 \cdot 2) = 570$	$570/16 = \mathbf{35.625}$
2	1	80	$\begin{matrix} 40 & 50 & 60 \\ 70 & \mathbf{80} & 90 \\ 0 & 0 & 0 \end{matrix}$	$(40 \cdot 1) + (50 \cdot 2) + (60 \cdot 1) + (70 \cdot 2) + (80 \cdot 4) + (90 \cdot 2) = 840$	$840/16 = \mathbf{52.500}$
2	2	90	$\begin{matrix} 50 & 60 & 0 \\ 80 & \mathbf{90} & 0 \\ 0 & 0 & 0 \end{matrix}$	$(50 \cdot 1) + (60 \cdot 2) + (80 \cdot 2) + (90 \cdot 4) = 690$	$690/16 = \mathbf{43.125}$

The **Final Filtered Image (O)** is:

$$O = \begin{pmatrix} 13.125 & 22.500 & 20.625 \\ 32.500 & 50.000 & 42.500 \\ 35.625 & 52.500 & 43.125 \end{pmatrix}$$

Given a 256x256 image with a maximum frequency of 128 Hz, a high-pass filter is applied to block frequencies lower than 50 Hz. Calculate the fraction of the frequency components that remain in percentage.

Step 1: Determine the Remaining Frequency Range

- Maximum frequency, $F_{\text{max}} = 128 \text{ Hz}$.
- High-pass filter blocks frequencies lower than 50 Hz.
- The high-pass filter **passes** all frequencies from 50 Hz up to the maximum.

$$\text{Remaining Range} = F_{\text{max}} - F_{\text{block}} = 128 \text{ Hz} - 50 \text{ Hz} = 78 \text{ Hz}$$

Step 2: Calculate the Fraction and Percentage

The total frequency range is from 0 Hz to 128 Hz, covering 128 Hz.

$$\text{Fraction Remaining} = \frac{\text{Remaining Range}}{\text{Total Range}} = \frac{78}{128}$$

$$\text{Fraction Remaining} = 0.609375$$

$$\text{Percentage Remaining} = 0.609375 \times 100 = \mathbf{60.9375\%}$$

The fraction of the frequency components that remain is 60.9375%.

An image has a size of 512x512. Using FFT for frequency domain filtering, calculate the number of operations required for the transformation, assuming NlogN complexity, where N=512×512

Step 1: Calculate the Total Number of Pixels (N)

The image size is 512×512.

$$N = 512 \times 512 = 262,144 \text{ pixels}$$

Step 2: Determine $\log_2 N$

Since $512 = 2^9$, the total number of pixels N is:

$$N = 512^2 = (2^9)^2 = 2^{18}$$

$$\log_2 N = 18$$

Step 3: Calculate the Number of Operations

The complexity for the 2D FFT transformation is proportional to $O(N \log N)$.

$$\text{Operations} \approx N \log_2 N$$

$$\text{Operations} \approx 262,144 \times 18 = \mathbf{4,718,592}$$

The number of operations required is approximately $4,718,592$.

Apply the Sobel operator on the following 3x3 matrix in the x-direction:
[100 150 200
100 150 200

100 150 200]

Step 1: Define the Sobel G_x Kernel

The Sobel operator for the x-direction (horizontal differentiation) is:

$$G_x = \begin{pmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{pmatrix}$$

Step 2: Apply Zero-Padding

To calculate a gradient value for all 9 original pixels using a 3×3 kernel, the 3×3 image patch must be zero-padded by a 1-pixel border.

$$\text{Padded Image } (I_{\text{padded}}) = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 100 & 150 & 200 & 0 \\ 0 & 100 & 150 & 200 & 0 \\ 0 & 100 & 150 & 200 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

Step 3: Perform Convolution for All 9 Pixels

The output $O(i, j)$ is the sum of the element-wise product of the 3×3 kernel G_x and the corresponding 3×3 window (W) in the padded image.

$$O(i, j) = \sum_{m=-1}^{1} \sum_{n=-1}^{1} W(m, n) \cdot G_x(m, n)$$

Row (i)	Col (j)	Original Pixel	Weighted Sum (Calculation)	Filtered Value
0	0	100	$(-1 \cdot 0) + (1 \cdot 0) + (-2 \cdot 0) + (2 \cdot 150) + (-1 \cdot 0) + (1 \cdot 150)$	$\mathbf{450}$
0	1	150	$(-1 \cdot 100) + (1 \cdot 200) + (-2 \cdot 100) + (2 \cdot 200) + (-1 \cdot 100) + (1 \cdot 200)$	$\mathbf{300}$
0	2	200	$(-1 \cdot 150) + (1 \cdot 0) + (-2 \cdot 150) + (2 \cdot 0) + (-1 \cdot 150) + (1 \cdot 0)$	$\mathbf{-450}$
1	0	100	$(-1 \cdot 0) + (1 \cdot 150) + (-2 \cdot 0) + (2 \cdot 150) + (-1 \cdot 0) + (1 \cdot 150)$	$\mathbf{600}$
1	1	150	$(-1 \cdot 100) + (1 \cdot 200) + (-2 \cdot 100) + (2 \cdot 200) + (-1 \cdot 100) + (1 \cdot 200)$	$\mathbf{400}$
1	2	200	$(-1 \cdot 150) + (1 \cdot 0) + (-2 \cdot 150) + (2 \cdot 0) + (-1 \cdot 150) + (1 \cdot 0)$	$\mathbf{-600}$
2	0	100	$(-1 \cdot 0) + (1 \cdot 150) + (-2 \cdot 0) + (2 \cdot 150) + (-1 \cdot 0) + (1 \cdot 0)$	$\mathbf{450}$
2	1	150	$(-1 \cdot 100) + (1 \cdot 200) + (-2 \cdot 100) + (2 \cdot 200) + (-1 \cdot 0) + (1 \cdot 0)$	$\mathbf{300}$
2	2	200	$(-1 \cdot 150) + (1 \cdot 0) + (-2 \cdot 150) + (2 \cdot 0) + (-1 \cdot 0) + (1 \cdot 0)$	$\mathbf{-450}$

The resulting **Sobel G_x Filtered Matrix** is:

$$O = \begin{pmatrix} 450 & 300 & -450 \\ 600 & 400 & -600 \\ 450 & 300 & -450 \end{pmatrix}$$

Given the following descriptors from two images, use FLANN to find the nearest neighbor for each descriptor in D_A from D_B. Apply Lowe's ratio test with a threshold of 0.45 and indicate whether the match is valid.
Descriptors for Image A (D_A):
0.1 0.5 0.3
0.4 0.2 0.8
0.6 0.7 0.9

Descriptors for Image B (D_B):
0.2 0.5 0.7
0.4 0.1 0.6
0.3 0.7 0.8
Euclidean Distance Formula:d= SQRT(∑(D _A [i]−D_B[j]) 2)
Find the nearest and second nearest neighbor for each descriptor in D_A.
Check the matches using Lowe's ratio test.

Step 1: Calculate the Distance Matrix

The Euclidean distance d between descriptor i from D_A and descriptor j from D_B is:

$$d_{ij} = \sqrt{\sum_{k=1}^{3}(D_A[i]_k - D_B[j]_k)^2}$$

The resulting distance matrix D (3×3) is (rounded to 4 decimal places):

$$D = \begin{pmatrix} d_{11} & d_{12} & d_{13} \\ d_{21} & d_{22} & d_{23} \\ d_{31} & d_{32} & d_{33} \end{pmatrix} = \begin{pmatrix} 0.4123 & 0.5831 & 0.5745 \\ 0.3742 & 0.2236 & 0.5099 \\ 0.4899 & 0.7000 & 0.3162 \end{pmatrix}$$

Step 2: Find Nearest Neighbors and Apply Lowe's Ratio Test

For each descriptor in D_A, we find its nearest neighbor (d_{nearest}) and second nearest neighbor ($d_{\text{second\_nearest}}$) in D_B. We then apply Lowe's ratio test: **Match is Valid if $R < 0.45$.**

$$\text{Ratio } (R) = \frac{d_{\text{nearest}}}{d_{\text{second\_nearest}}}$$

D_A Descriptor	d_{nearest} (Index D_B)	$d_{\text{second\_nearest}}$ (Index D_B)	Ratio (R)	Valid Match ($R < 0.45$)?
1 (0.1, 0.5, 0.3)	0.4123 (B_1)	0.5745 (B_3)	0.7177	**False**
2 (0.4, 0.2, 0.8)	0.2236 (B_2)	0.3742 (B_1)	0.5976	**False**
3 (0.6, 0.7, 0.9)	0.3162 (B_3)	0.4899 (B_1)	0.6455	**False**

3. Harris Corner Detection

Given Image I:

$$I = \begin{pmatrix} 1 & 2 & 3 \\ 4 & \mathbf{5} & 6 \\ 7 & 8 & 9 \end{pmatrix}$$

a. Compute the Gradients (I_x and I_y)

We apply the Sobel operator to the center pixel $I(1,1) = 5$:

$$\mathbf{I_x} = (-1 \cdot 1) + (1 \cdot 3) + (-2 \cdot 4) + (2 \cdot 6) + (-1 \cdot 7) + (1 \cdot 9)$$

$$I_x = -1 + 3 - 8 + 12 - 7 + 9 = \mathbf{8}$$

$$\mathbf{I_y} = (-1 \cdot 1) + (-2 \cdot 2) + (-1 \cdot 3) + (1 \cdot 7) + (2 \cdot 8) + (1 \cdot 9)$$

$$I_y = -1 - 4 - 3 + 7 + 16 + 9 = \mathbf{24}$$

b. Calculate the Structure Tensor M

Assuming a 1×1 window (no summation):

$$M = \begin{pmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{pmatrix} = \begin{pmatrix} 8^2 & 8 \cdot 24 \\ 8 \cdot 24 & 24^2 \end{pmatrix}$$

$$M = \begin{pmatrix} \mathbf{64} & \mathbf{192} \\ \mathbf{192} & \mathbf{576} \end{pmatrix}$$

c. Compute the Harris Corner Response R

$$R = \det(M) - k \cdot (\text{trace}(M))^2, \quad \text{where } k = 0.04$$

1. **Determinant:** $\det(M) = (I_x^2)(I_y^2) - (I_x I_y)^2 = 64 \cdot 576 - 192^2 = 36864 - 36864 = \mathbf{0}$
2. **Trace:** $\text{trace}(M) = I_x^2 + I_y^2 = 64 + 576 = \mathbf{640}$

$$R = 0 - 0.04 \cdot (640)^2 = -0.04 \cdot 409600$$

$$R = -\mathbf{16384.00}$$

d. Classify the Pixel at (1, 1)

- The determinant of M is $\det(M) = 0$, which implies that one of the eigenvalues is zero ($\lambda_2 = 0$), meaning the gradient is constant in one direction.
- The large negative value for R confirms this.

Classification: The pixel at (1, 1) is classified as a **Non-interest Point** or, more specifically, a **Flat Region or Edge**. Since the gradient (M=640) is non-zero, it is part of a strong **Edge** (a line with constant intensity change)