# Write a program to display image matrix

import cv2

import numpy as np

image = cv2.imread("/content/letter-7-transformed.png")

image\_matrix = np.array(image)

print("Image Matrix:")

print(image\_matrix)

Image Matrix:

[[[14 14 14]

[ 0 0 0]

[ 0 0 0]

...

[ 0 0 0]

[ 0 0 0]

[ 0 0 0]]

[[14 14 14]

[ 0 0 0]

[ 0 0 0]

...

[ 0 0 0]

[ 0 0 0]

[ 0 0 0]]

[[14 14 14]

[ 0 0 0]

[ 0 0 0]

...

[ 0 0 0]

[ 0 0 0]

[ 0 0 0]]

...

[[14 14 14]

[ 0 0 0]

[ 0 0 0]

...

[ 0 0 0]

[ 0 0 0]

[ 0 0 0]]

[[14 14 14]

[ 0 0 0]

[ 0 0 0]

...

[ 0 0 0]

[ 0 0 0]

[ 0 0 0]]

[[14 14 14]

[ 0 0 0]

[ 0 0 0]

...

[ 0 0 0]

[ 0 0 0]

[ 0 0 0]]]

# 2. write program to display Image Histogram

import cv2

import matplotlib.pyplot as plt

image\_path = "/content/letter-7-transformed.png"

image = cv2.imread(image\_path, cv2.IMREAD\_GRAYSCALE)

histogram = cv2.calcHist([image], [0], None, [256], [0, 256])

plt.figure(figsize=(8, 6))

plt.title("Image Histogram")

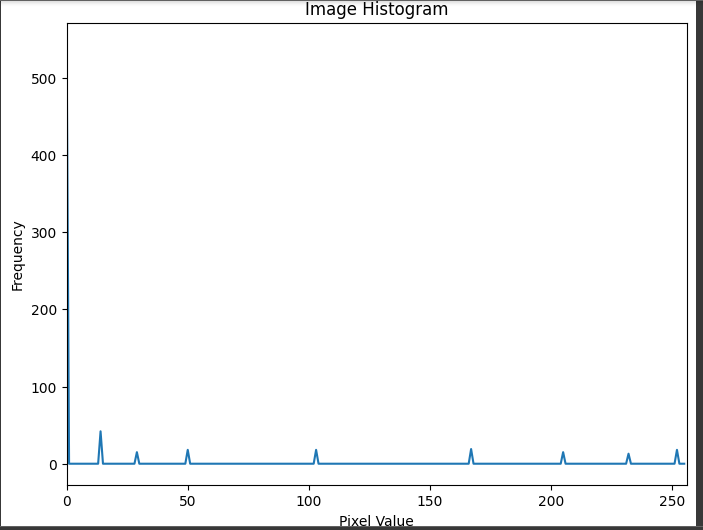
plt.xlabel("Pixel Value")

plt.ylabel("Frequency")

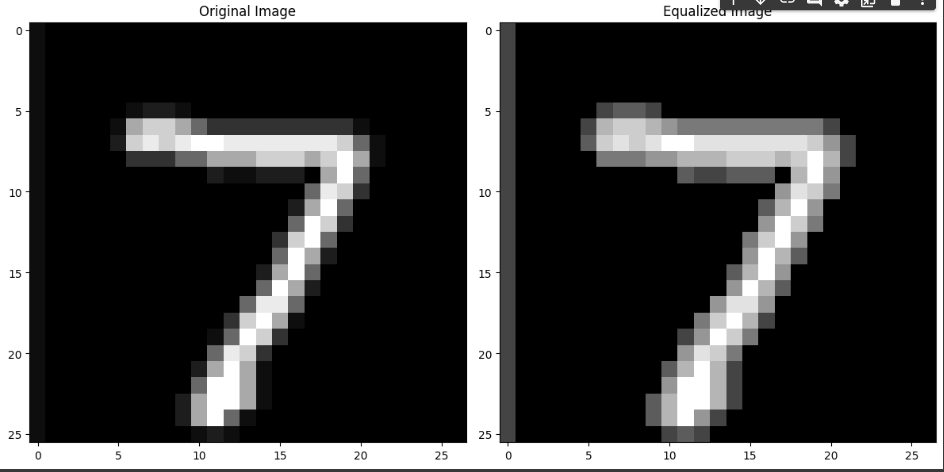
plt.plot(histogram)

plt.xlim([0, 256])

plt.show()



1. write program to find Histogram equalization
2. import cv2
3. import numpy as np
4. import matplotlib.pyplot as plt
5. image\_path = "/content/letter-7-transformed.png"
6. image = cv2.imread(image\_path, cv2.IMREAD\_GRAYSCALE)
7. equalized\_image = cv2.equalizeHist(image)
8. plt.figure(figsize=(12, 6))
9. plt.subplot(1, 2, 1)
10. plt.title("Original Image")
11. plt.imshow(image, cmap='gray')
12. plt.subplot(1, 2, 2)
13. plt.title("Equalized Image")
14. plt.imshow(equalized\_image, cmap='gray')
15. plt.tight\_layout()
16. plt.show()



4. write program to smooth image using gaussian filter

import cv2

import numpy as np

import matplotlib.pyplot as plt

image\_path = "/content/letter-7-transformed.png"

image = cv2.imread(image\_path)

smoothed\_image = cv2.GaussianBlur(image, (5, 5), 0)

plt.figure(figsize=(12, 6))

plt.subplot(1, 2, 1)

plt.title("Original Image")

plt.imshow(cv2.cvtColor(image, cv2.COLOR\_BGR2RGB))

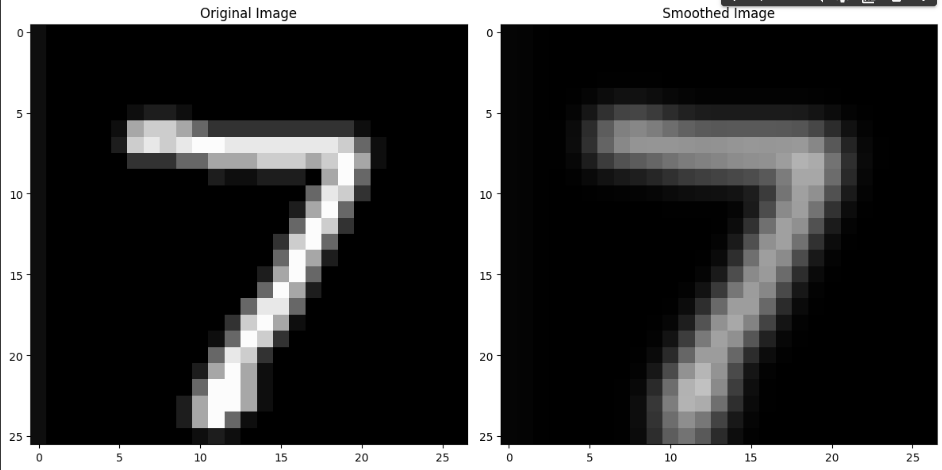
plt.subplot(1, 2, 2)

plt.title("Smoothed Image")

plt.imshow(cv2.cvtColor(smoothed\_image, cv2.COLOR\_BGR2RGB))

plt.tight\_layout()

plt.show()



5. write program to find 1st order dertivative and display image

import cv2

import numpy as np

import matplotlib.pyplot as plt

image\_path = "/content/letter-7-transformed.png"

image = cv2.imread(image\_path, cv2.IMREAD\_GRAYSCALE)

dx = cv2.Sobel(image, cv2.CV\_64F, 1, 0, ksize=3)

dy = cv2.Sobel(image, cv2.CV\_64F, 0, 1, ksize=3)

plt.figure(figsize=(12, 6))

plt.subplot(1, 3, 1)

plt.title("Original Image")

plt.imshow(image, cmap='gray')

plt.subplot(1, 3, 2)

plt.title("X-Gradient")

plt.imshow(dx, cmap='gray')

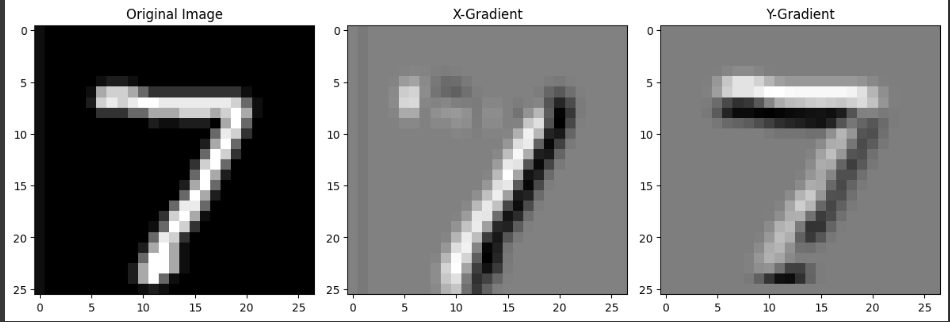
plt.subplot(1, 3, 3)

plt.title("Y-Gradient")

plt.imshow(dy, cmap='gray')

plt.tight\_layout()

plt.show()



6. write program to find second order derivative and display image

import cv2

import numpy as np

import matplotlib.pyplot as plt

image\_path = "/content/letter-7-transformed.png"

image = cv2.imread(image\_path, cv2.IMREAD\_GRAYSCALE)

laplacian = cv2.Laplacian(image, cv2.CV\_64F)

plt.figure(figsize=(12, 6))

plt.subplot(1, 2, 1)

plt.title("Original Image")

plt.imshow(image, cmap='gray')

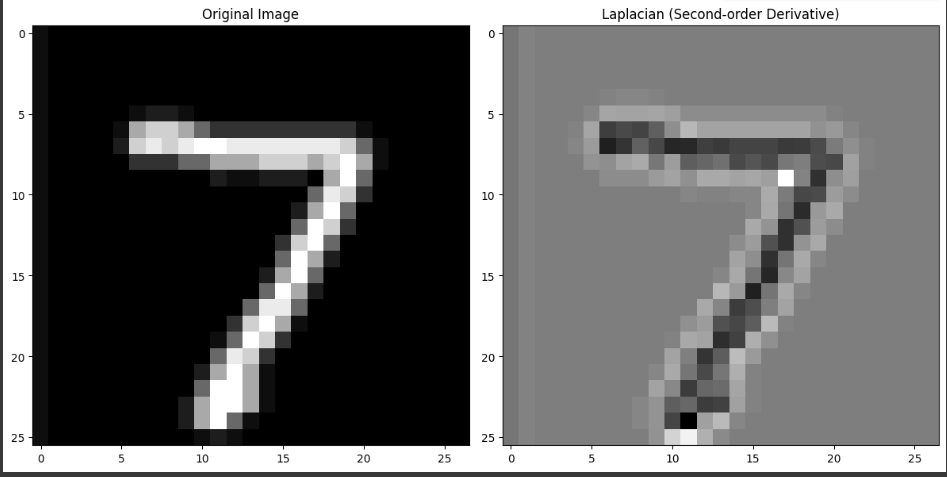
plt.subplot(1, 2, 2)

plt.title("Laplacian (Second-order Derivative)")

plt.imshow(laplacian, cmap='gray')

plt.tight\_layout()

plt.show()



7. write program to determine Image gradient using Sobel operators

import cv2

import numpy as np

import matplotlib.pyplot as plt

image\_path = "/content/letter-7-transformed.png"

image = cv2.imread(image\_path, cv2.IMREAD\_GRAYSCALE)

gradient\_x = cv2.Sobel(image, cv2.CV\_64F, 1, 0, ksize=3)

gradient\_y = cv2.Sobel(image, cv2.CV\_64F, 0, 1, ksize=3)

gradient\_magnitude = np.sqrt(gradient\_x\*\*2 + gradient\_y\*\*2)

gradient\_direction = np.arctan2(gradient\_y, gradient\_x)

plt.figure(figsize=(15, 8))

plt.subplot(2, 3, 1)

plt.title("Original Image")

plt.imshow(image, cmap='gray')

plt.subplot(2, 3, 2)

plt.title("X-Gradient")

plt.imshow(gradient\_x, cmap='gray')

plt.subplot(2, 3, 3)

plt.title("Y-Gradient")

plt.imshow(gradient\_y, cmap='gray')

plt.subplot(2, 3, 4)

plt.title("Gradient Magnitude")

plt.imshow(gradient\_magnitude, cmap='gray')

plt.subplot(2, 3, 5)

plt.title("Gradient Direction")

plt.imshow(gradient\_direction, cmap='gray')

plt.tight\_layout()

plt.show()

