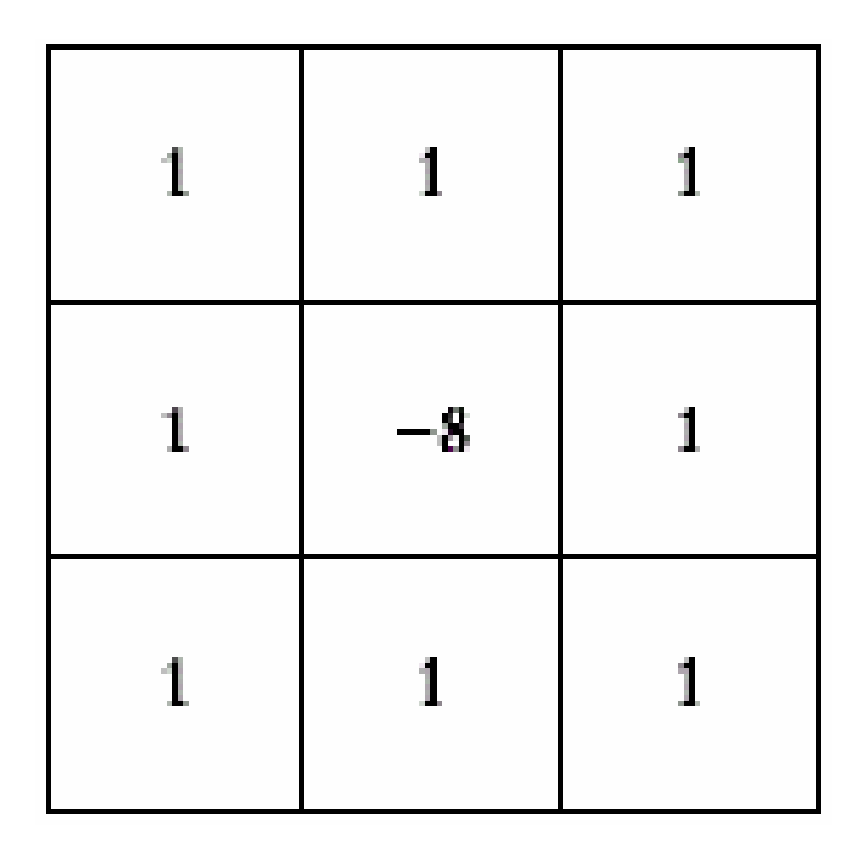
**21-30 Programs**

**21. Perform Sharpening of Image using Laplacian mask implemented with an extension of**

**diagonal neighbors,**

****

import cv2

import numpy as np

from matplotlib import pyplot as plt

# Read the image in grayscale

img = cv2.imread("C:\\Users\\user\\OneDrive\\Documents\\Computer vision with openCV\\sample image.jpg", cv2.IMREAD\_GRAYSCALE)

# Extended Laplacian kernel including diagonal neighbors

laplacian\_ext = np.array([[1, 1, 1],

[1, -8, 1],

[1, 1, 1]])

# Apply convolution

sharpened = cv2.filter2D(img, -1, laplacian\_ext)

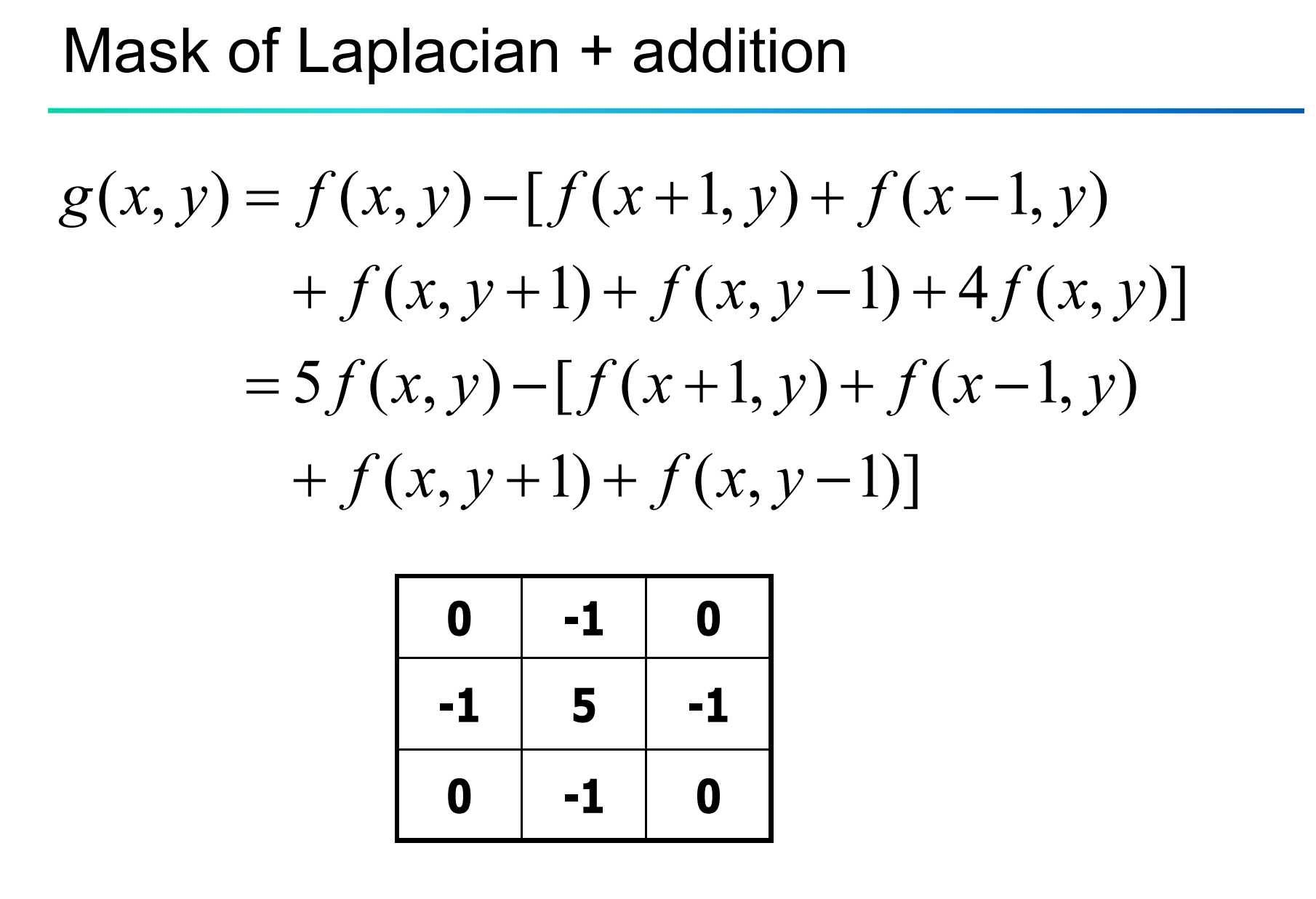
# Show results

plt.subplot(1,2,1), plt.title("Original"), plt.imshow(img, cmap='gray'), plt.axis('off')

plt.subplot(1,2,2), plt.title("Sharpened"), plt.imshow(sharpened, cmap='gray'), plt.axis('off')

plt.tight\_layout(), plt.show()

**22. Perform Sharpening of Image using Laplacian mask with positive center coefficient.**

****

import cv2

import numpy as np

from matplotlib import pyplot as plt

# Read the image in grayscale

img = cv2.imread("C:\\Users\\user\\OneDrive\\Documents\\Computer vision with openCV\\sample image.jpg", cv2.IMREAD\_GRAYSCALE)

# Laplacian mask with positive center (5) and -1 at four neighbors

laplacian\_positive = np.array([[ 0, -1, 0],

[-1, 5, -1],

[ 0, -1, 0]])

# Apply the filter

sharpened = cv2.filter2D(img, -1, laplacian\_positive)

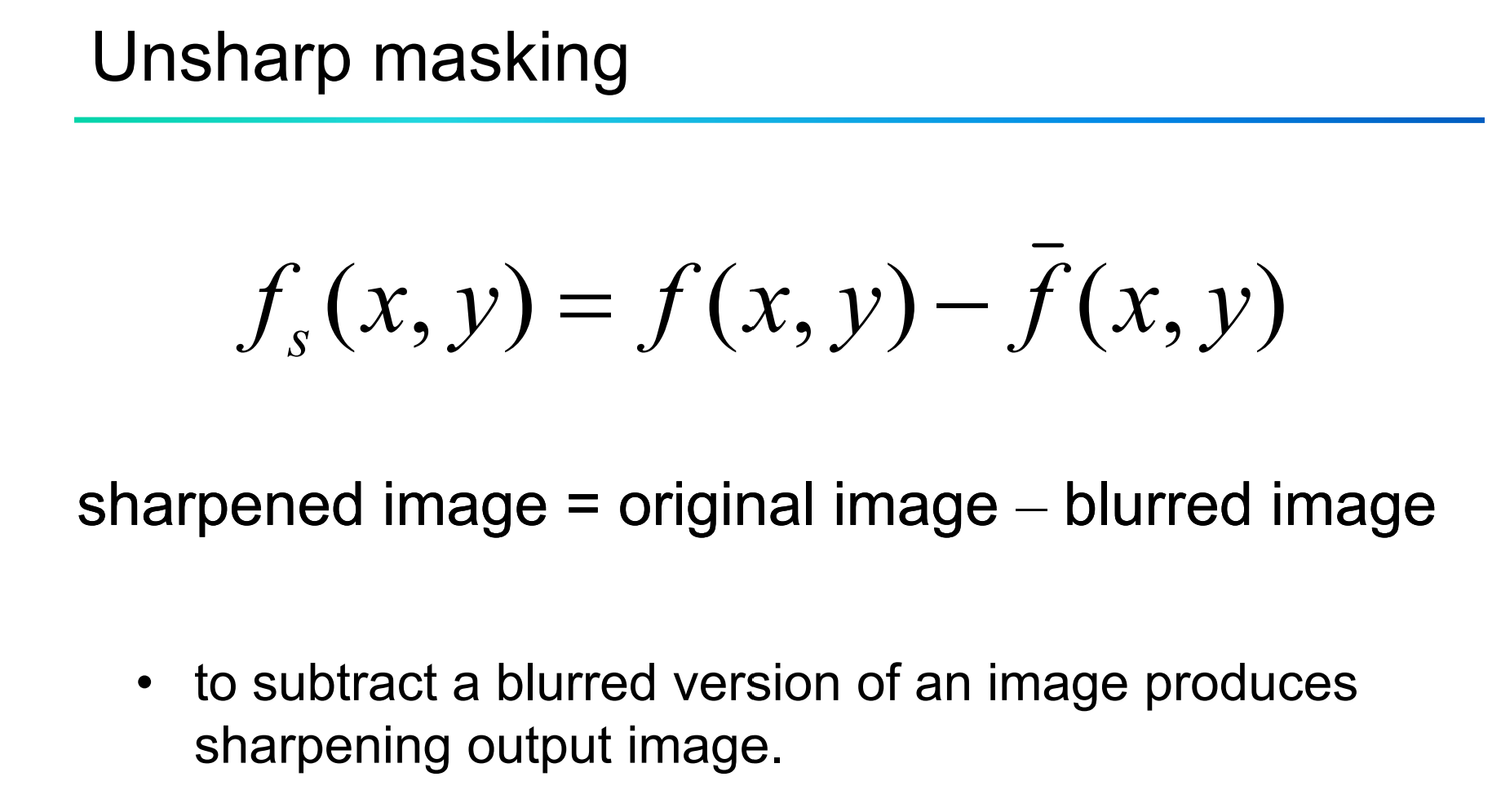
# Show results

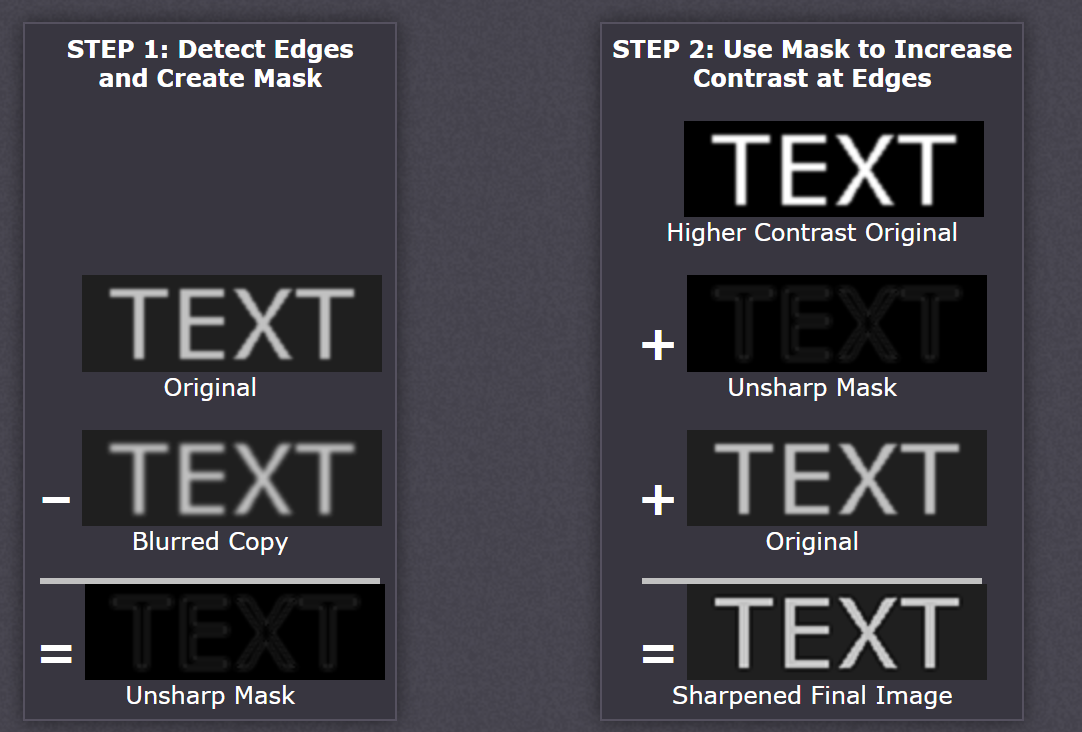
plt.subplot(1,2,1), plt.title("Original"), plt.imshow(img, cmap='gray'), plt.axis('off')

plt.subplot(1,2,2), plt.title("Sharpened"), plt.imshow(sharpened, cmap='gray'), plt.axis('off')

plt.tight\_layout(), plt.show()

**23. Perform Sharpening of Image using unsharp masking.**

****

****

import cv2

import numpy as np

from matplotlib import pyplot as plt

# Read the image in grayscale

img = cv2.imread("C:\\Users\\user\\OneDrive\\Documents\\Computer vision with openCV\\sample image.jpg", cv2.IMREAD\_GRAYSCALE)

# Laplacian mask with positive center (5) and -1 at four neighbors

laplacian\_positive = np.array([[ 0, -1, 0],

[-1, 5, -1],

[ 0, -1, 0]])

# Apply the filter

sharpened = cv2.filter2D(img, -1, laplacian\_positive)

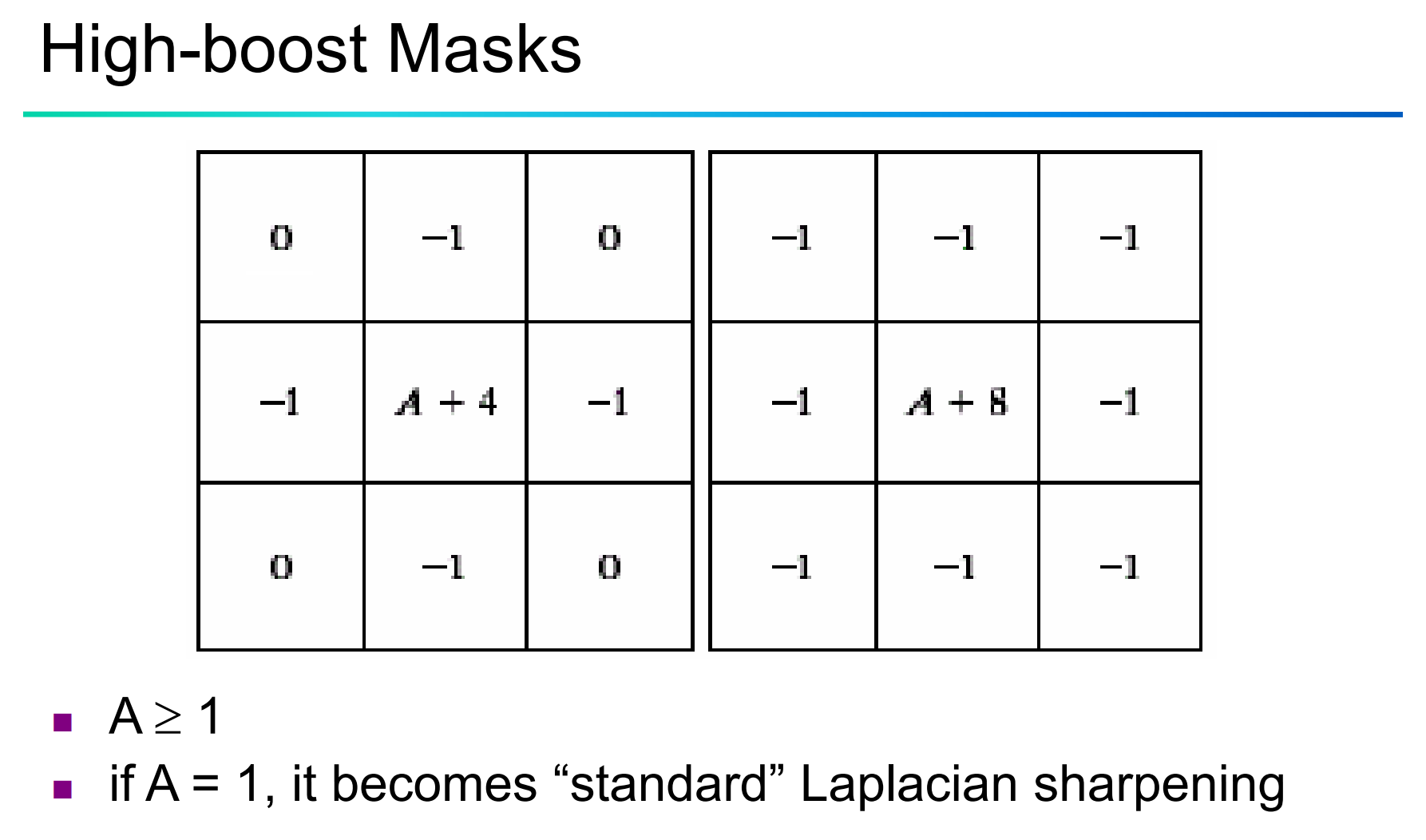
# Show results

plt.subplot(1,2,1), plt.title("Original"), plt.imshow(img, cmap='gray'), plt.axis('off')

plt.subplot(1,2,2), plt.title("Sharpened"), plt.imshow(sharpened, cmap='gray'), plt.axis('off')

plt.tight\_layout(), plt.show()

**24. Perform Sharpening of Image using High-Boost Masks.**

****

import cv2

import numpy as np

import matplotlib.pyplot as plt

# Read image in grayscale

img = cv2.imread("C:\\Users\\user\\OneDrive\\Documents\\Computer vision with openCV\\image1.png", cv2.IMREAD\_GRAYSCALE)

# Choose A ≥ 1 (e.g., A = 1.5 for high-boost)

A = 1.5

# High-boost kernel (center = A + 8, others = -1)

high\_boost\_kernel = np.array([[-1, -1, -1],

[-1, A + 8, -1],

[-1, -1, -1]])

# Apply filter

sharpened = cv2.filter2D(img, -1, high\_boost\_kernel)

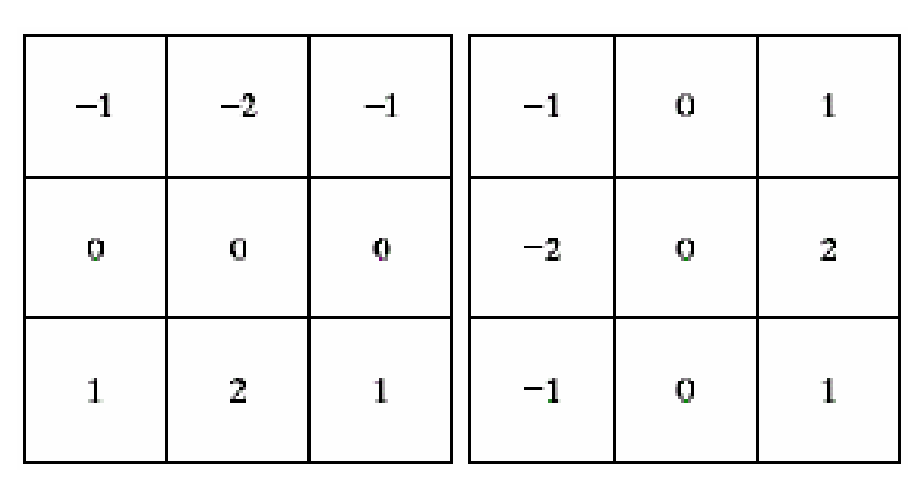
# Display result

plt.subplot(1,2,1), plt.title("Original"), plt.imshow(img, cmap='gray'), plt.axis('off')

plt.subplot(1,2,2), plt.title("High-Boost Sharpened"), plt.imshow(sharpened, cmap='gray'), plt.axis('off')

plt.tight\_layout(), plt.show()

**25. Perform Sharpening of Image using Gradient masking.**

****

import cv2

import numpy as np

import matplotlib.pyplot as plt

# Read image in grayscale

img = cv2.imread("C:\\Users\\user\\OneDrive\\Documents\\Computer vision with openCV\\image2.jpg", cv2.IMREAD\_GRAYSCALE)

# Define gradient masks (Prewitt-like)

gx = np.array([[-1, 0, 1],

[-2, 0, 2],

[-1, 0, 1]])

gy = np.array([[-1, -2, -1],

[ 0, 0, 0],

[ 1, 2, 1]])

# Apply convolution

grad\_x = cv2.filter2D(img, -1, gx)

grad\_y = cv2.filter2D(img, -1, gy)

# Combine gradients (magnitude)

sharpened = cv2.addWeighted(grad\_x, 0.5, grad\_y, 0.5, 0)

# Display results

plt.subplot(1,2,1), plt.title("Original"), plt.imshow(img, cmap='gray'), plt.axis('off')

plt.subplot(1,2,2), plt.title("Gradient Sharpened"), plt.imshow(sharpened, cmap='gray'), plt.axis('off')

plt.tight\_layout(), plt.show()

**26. Insert water marking to the image using OpenCV.**

import cv2

# Load the image

img = cv2.imread("C:\\Users\\user\\OneDrive\\Documents\\Computer vision with openCV\\image3.jpg")

# Add watermark text

cv2.putText(img, 'Watermark', (10, 30), cv2.FONT\_HERSHEY\_SIMPLEX,

1, (255, 255, 255), 2, cv2.LINE\_AA)

# Save or display the watermarked image

cv2.imwrite("C:\\Users\\user\\OneDrive\\Documents\\Computer vision with openCV\\watermark.png", img)

cv2.imshow('Watermarked', img)

cv2.waitKey(0)

cv2.destroyAllWindows()

**27. Do Cropping, Copying and pasting image inside another image using OpenCV.**

import cv2

# Load image

img = cv2.imread("C:\\Users\\user\\OneDrive\\Documents\\Computer vision with openCV\\image5.jpg")

# Crop a region (y1:y2, x1:x2)

crop = img[50:150, 100:200]

# Get height and width of the cropped region

h, w, \_ = crop.shape

# Make sure the destination fits inside the original image

if img.shape[0] > 200 + h and img.shape[1] > 300 + w:

# Paste the cropped region at new location

img[200:200+h, 300:300+w] = crop

else:

print("Destination region is out of bounds!")

# Show and save result

cv2.imshow('Pasted Image', img)

cv2.imwrite('pasted\_output.png', img)

cv2.waitKey(0)

cv2.destroyAllWindows()

**28. Find the boundary of the image using Convolution kernel for the given image.**

import cv2

import numpy as np

import matplotlib.pyplot as plt

# Load image in grayscale

img = cv2.imread("C:\\Users\\user\\OneDrive\\Documents\\Computer vision with openCV\\image2.jpg", cv2.IMREAD\_GRAYSCALE)

# Define a simple edge-detection kernel (Laplacian)

kernel = np.array([[0, -1, 0],

[-1, 4, -1],

[0, -1, 0]])

# Apply convolution

edges = cv2.filter2D(img, -1, kernel)

# Show result

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

plt.title('Image Boundary'), plt.axis('off')

plt.show()

**29. Morphological operations based on OpenCV using Erosion technique.**

import cv2

import numpy as np

# Read image in grayscale

img = cv2.imread("C:\\Users\\user\\OneDrive\\Documents\\Computer vision with openCV\\image3.jpg", 0)

# Define a kernel

kernel = np.ones((5,5), np.uint8)

# Apply erosion

eroded = cv2.erode(img, kernel, iterations=1)

# Show result

cv2.imshow('Eroded Image', eroded)

cv2.waitKey(0)

cv2.destroyAllWindows()

**30. Morphological operations based on OpenCV using Dilation technique.**

import cv2

import numpy as np

# Read image in grayscale

img = cv2.imread("C:\\Users\\user\\OneDrive\\Documents\\Computer vision with openCV\\image5.jpg", 0)

# Define kernel

kernel = np.ones((5,5), np.uint8)

# Apply dilation

dilated = cv2.dilate(img, kernel, iterations=1)

# Show result

cv2.imshow('Dilated Image', dilated)

cv2.waitKey(0)

cv2.destroyAllWindows()