****

**Lab Manual**

**Computer Engineering – Artificial Intelligence**

**B. Tech. Year – II, 5th Semester, Academic Year (2023)**

**Subject Code: 01AI0504**

*Subject Name: Digital Image Processing*

*Name: Basid Al Siddik Shammo*

*Enrollment Number: 92100151052*

**Aim-** Implementation of filters for smoothening images.

**Description-** The implementation of image smoothening filters involves applying specialized algorithms to reduce high-frequency noise and details, resulting in a softened and more visually pleasing appearance. By convolving these filters with the image, variations in pixel intensities are gently blended, effectively reducing sharp transitions and enhancing overall image cohesiveness. This process is commonly used in image processing and computer vision applications to improve image quality and aid in subsequent analysis or visual presentation.

**Task- I**

**Perform convolution with:**

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

[-1, 8, -1],

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

(i) Use 12x12 random matrix with maximum gray level 32.

(ii) Convolution with padding 1 and 2.

**Code-**

import numpy as np

from scipy.signal import convolve2d

import matplotlib.pyplot as plt

random\_matrix = np.random.randint(0, 33, size=(12, 12))

print("Random 12x12 Matrix:")

print(random\_matrix)

print("\n")

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

[-1,  8, -1],

                    [-1, -1, -1]])

convolution\_result = convolve2d(random\_matrix, kernel, mode='valid')

padded\_matrix\_1 = np.pad(random\_matrix, pad\_width=1, mode='constant', constant\_values=0)

convolution\_result\_padding\_1 = convolve2d(padded\_matrix\_1, kernel, mode='valid')

padded\_matrix\_2 = np.pad(random\_matrix, pad\_width=2, mode='constant', constant\_values=0)

convolution\_result\_padding\_2 = convolve2d(padded\_matrix\_2, kernel, mode='valid')

print("Convolution with Padding 1:")

print(convolution\_result\_padding\_1)

print("\n")

print("Convolution with Padding 2:")

print(convolution\_result\_padding\_2)

plt.figure(figsize=(10, 8))

plt.subplot(2, 2, 1)

plt.imshow(random\_matrix, cmap='gray')

plt.title('Original Random Matrix')

plt.subplot(2, 2, 2)

plt.imshow(convolution\_result, cmap='gray')

plt.title('Convolution Result (No Padding)')

plt.subplot(2, 2, 3)

plt.imshow(convolution\_result\_padding\_1, cmap='gray')

plt.title('Convolution Result (Padding 1)')

plt.subplot(2, 2, 4)

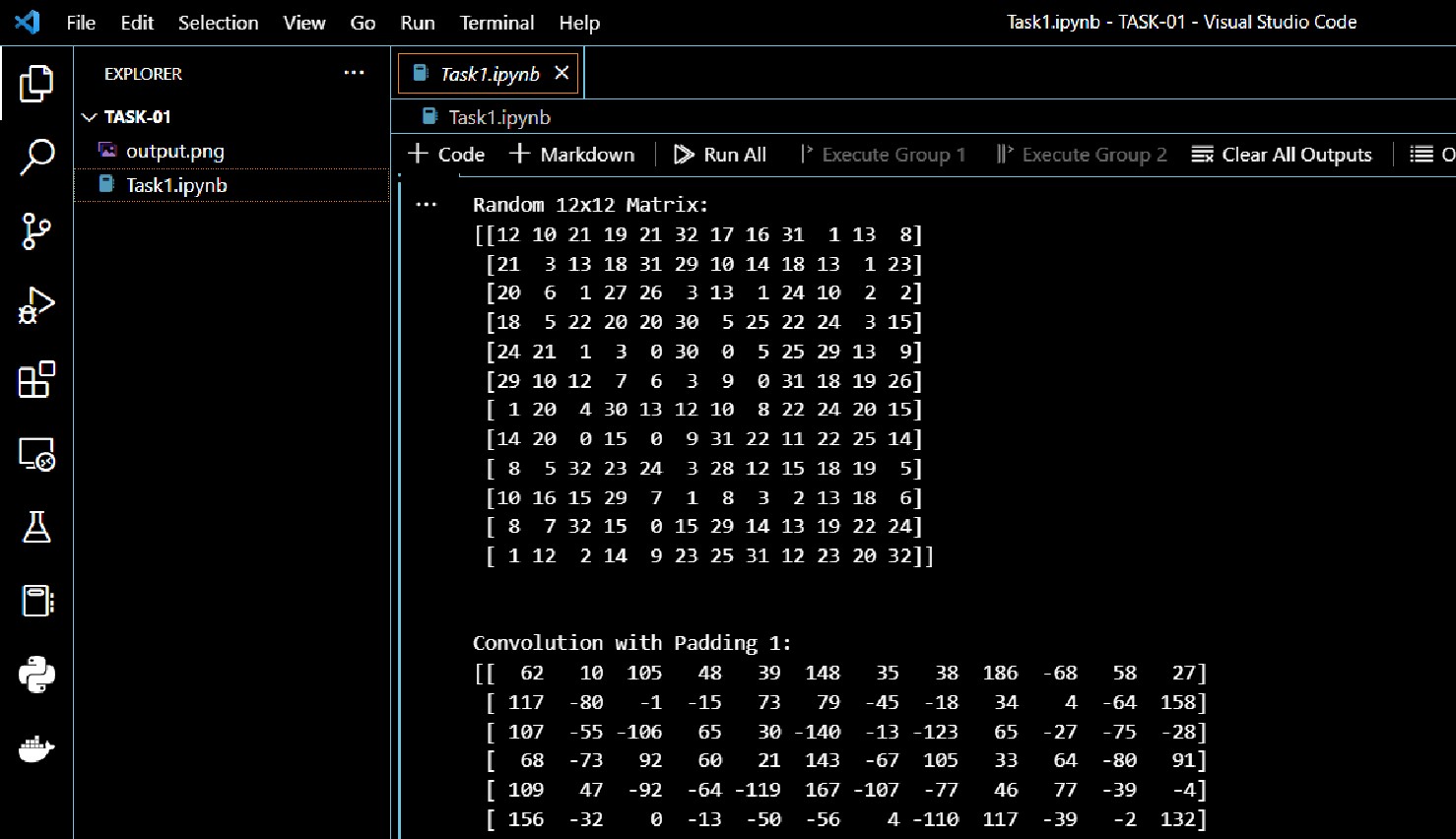
plt.imshow(convolution\_result\_padding\_2, cmap='gray')

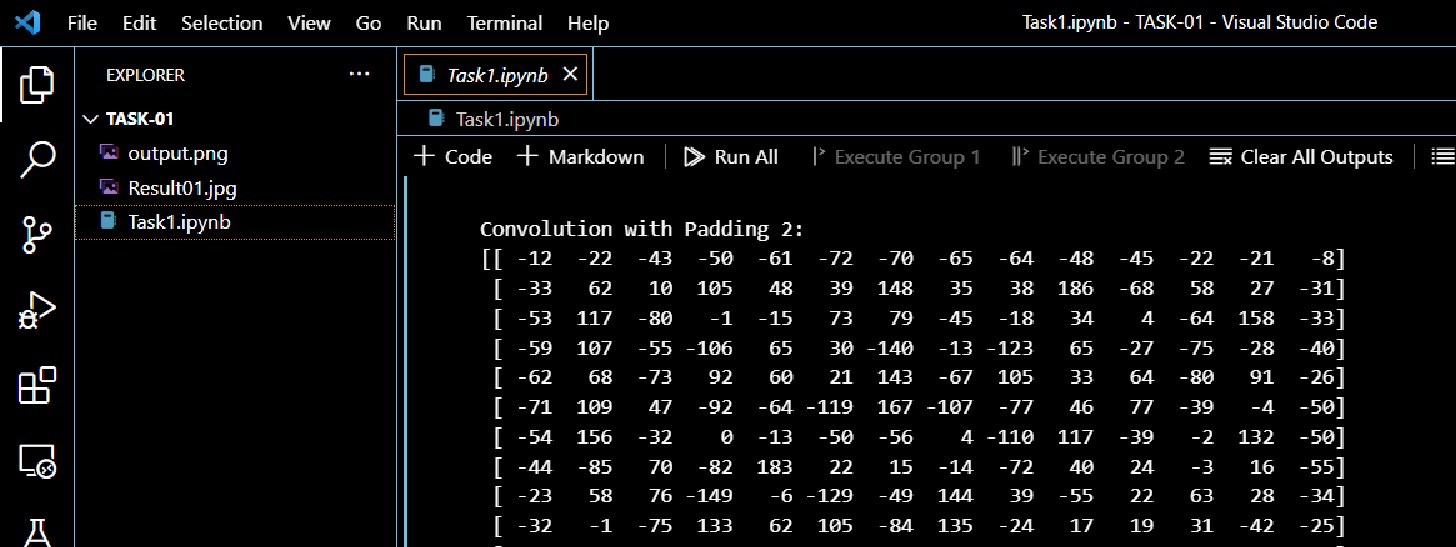
plt.title('Convolution Result (Padding 2)')

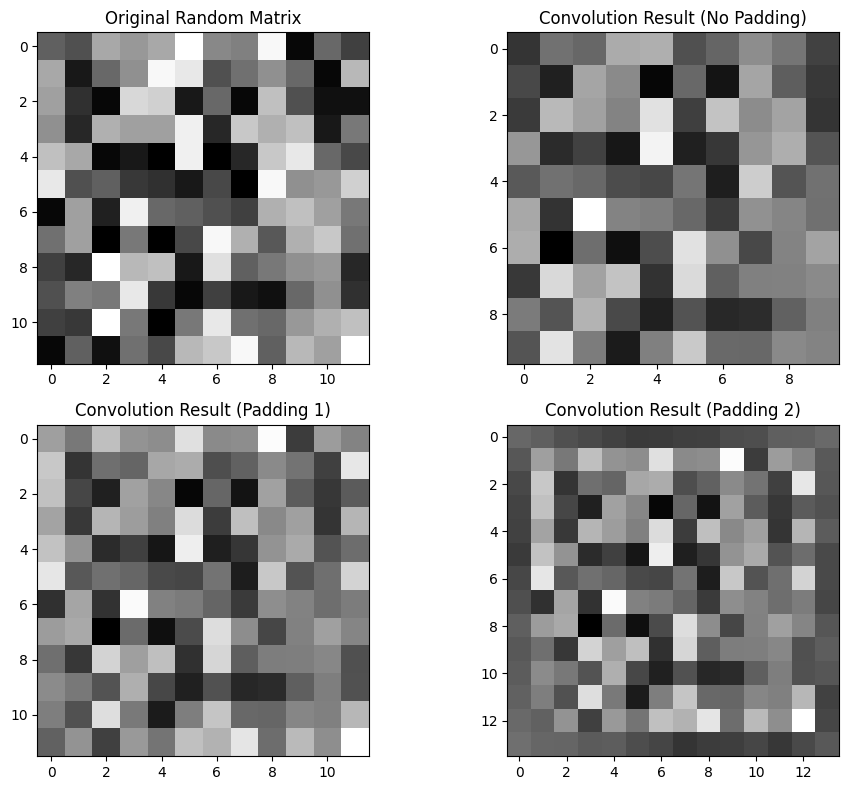
plt.tight\_layout()

plt.show()

**Result-**





****

**Task- II**

**Smoothen an image added with a noise with the following filters:**

(i) average (ii) median (iii) Box (iv) Gaussian.

**For noise addition you may use below:**

uni\_noise=np.zeros((640,480),dtype=np.uint8)

cv2.randu(uni\_noise,0,255)

uni\_noise=(uni\_noise\*0.5).astype(np.uint8)In [8]:

un\_img=cv2.add(img,uni\_noise)

**Code-**

import numpy as np

import cv2

from scipy.signal import convolve2d

import matplotlib.pyplot as plt

def apply\_filters(image):

    #i

    average\_filtered = cv2.blur(image, (5, 5))

    #ii

    median\_filtered = cv2.medianBlur(image, 5)

    #iii

    box\_kernel = np.ones((5, 5), np.float32) / 25

    box\_filtered = cv2.filter2D(image, -1, box\_kernel)

    #iv

    gaussian\_filtered = cv2.GaussianBlur(image, (5, 5), 0)

    return average\_filtered, median\_filtered, box\_filtered, gaussian\_filtered

img = np.random.randint(0, 256, size=(640, 480), dtype=np.uint8)

uni\_noise = np.zeros((640, 480), dtype=np.uint8)

cv2.randu(uni\_noise, 0, 255)

uni\_noise = (uni\_noise \* 0.5).astype(np.uint8)

uni\_img = cv2.add(img, uni\_noise)

averageg\_filtered, median\_filtered, box\_filtered, gaussian\_filtered = apply\_filters(uni\_img)

plt.figure(figsize=(12, 10))

plt.subplot(2, 3, 1)

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

plt.title('Original Image')

plt.subplot(2, 3, 2)

plt.imshow(uni\_img, cmap='gray')

plt.title('Image with Uniform Noise')

plt.subplot(2, 3, 3)

plt.imshow(averageg\_filtered, cmap='gray')

plt.title('Average Filter')

plt.subplot(2, 3, 4)

plt.imshow(median\_filtered, cmap='gray')

plt.title('Median Filter')

plt.subplot(2, 3, 5)

plt.imshow(box\_filtered, cmap='gray')

plt.title('Box Filter')

plt.subplot(2, 3, 6)

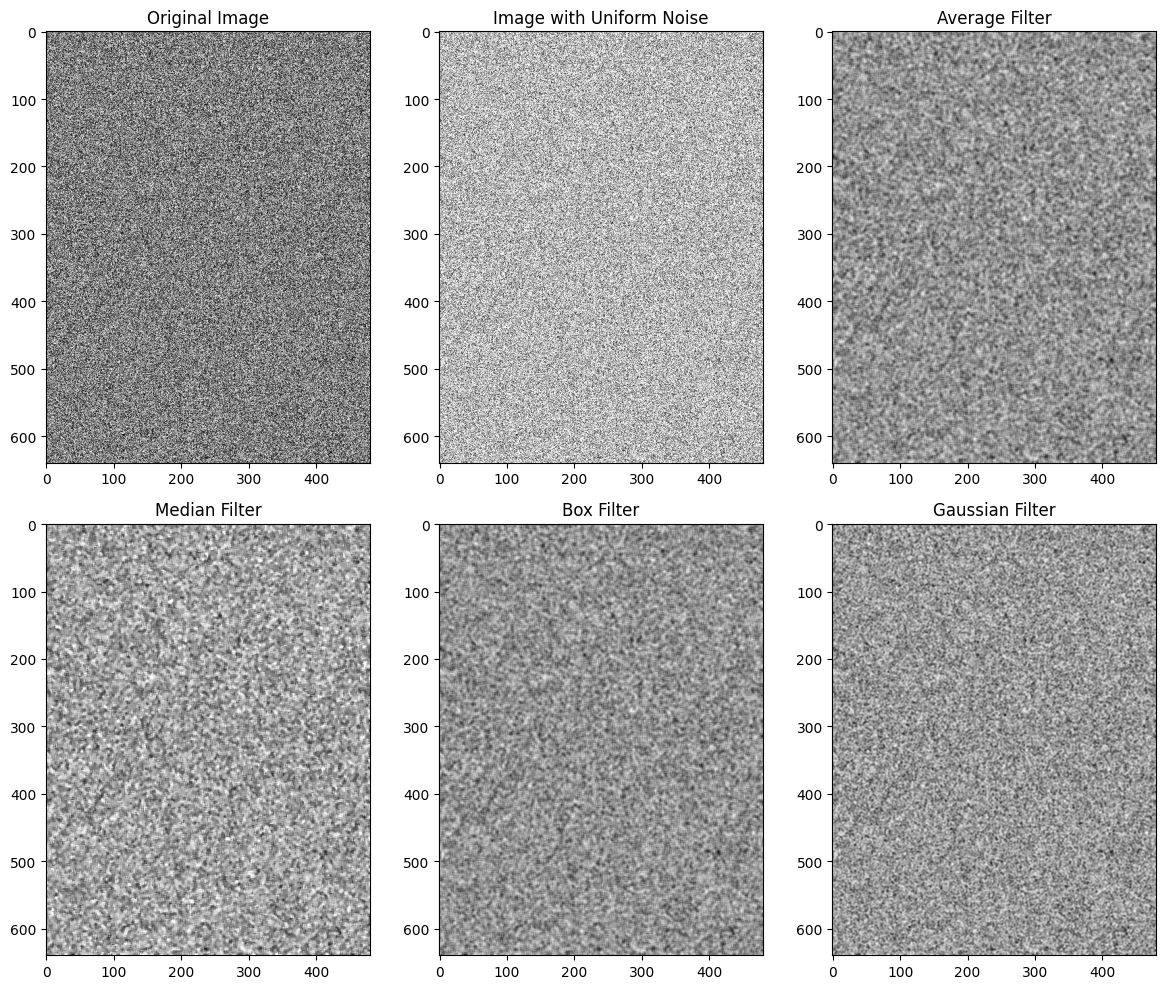
plt.imshow(gaussian\_filtered, cmap='gray')

plt.title('Gaussian Filter')

plt.tight\_layout()

plt.show()

**Result-**



**Conclusion-**