**COMPUTER VISION AND IMAGE ANALYSIS**

**Tasks**

**Task 1:** Perform the following steps:

Step 1: Read the input image (preferably 256 x 256 gray scale cameraman image)

Step 2: Blur the image by passing it through low pass filter. (Chose the mask in such a way that blurring effect is visible)

Step 3: Subtract the blurred version of the image from the input image and comment on the observed result.

import numpy as np

import cv2

import matplotlib.pyplot as plt

# *Step 1: Read the input image*

image = cv2.imread('cameraman.png', cv2.IMREAD\_GRAYSCALE)

# *Step 2: Blur the image using a low-pass filter*

kernel = np.ones((5, 5), np.float32) / 25

blurred\_image = cv2.filter2D(image, -1, kernel)

# *Step 3: Subtract the blurred version of the image from the input image*

high\_pass\_image = cv2.subtract(image, blurred\_image)

# *Display the images*

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

plt.subplot(1, 3, 1)

plt.title('Original Image')

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

plt.subplot(1, 3, 2)

plt.title('Blurred Image')

plt.imshow(blurred\_image, cmap='gray')

plt.subplot(1, 3, 3)

plt.title('High-pass Image (Original - Blurred)')

plt.imshow(high\_pass\_image, cmap='gray')

plt.show()

**Task 2**: Perform the following steps

Step 1: Read the input image (gray scale image)

Step 2: Add random noise to the input image.

Step 3: Pass the noisy image through high pass filter and comment on the observed result.

# *Step 1: Read the input image*

image = cv2.imread('cameraman.png', cv2.IMREAD\_GRAYSCALE)

# *Step 2: Add random noise to the input image*

noise = np.random.normal(0, 25, image.shape).astype(np.uint8)

noisy\_image = cv2.add(image, noise)

# *Step 3: Pass the noisy image through a high-pass filter*

# *Using a simple Laplacian filter for high-pass filtering*

high\_pass\_filter = np.array([[-1, -1, -1],

[-1, 8, -1],

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

high\_pass\_image = cv2.filter2D(noisy\_image, -1, high\_pass\_filter)

# *Display the images*

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

plt.subplot(1, 3, 1)

plt.title('Original Image')

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

plt.subplot(1, 3, 2)

plt.title('Noisy Image')

plt.imshow(noisy\_image, cmap='gray')

plt.subplot(1, 3, 3)

plt.title('High-pass Filtered Image')

plt.imshow(high\_pass\_image, cmap='gray')

plt.show()

**Task 3**: Write a python code to perform Vector Quantization of the input matrix as discussed in the class.

from sklearn.cluster import KMeans

def vector\_quantization(image, n\_clusters):

# *Reshape the image into a 2D array of pixels*

pixels = image.reshape(-1, 1)

# *Perform KMeans clustering*

kmeans = KMeans(n\_clusters=n\_clusters, random\_state=0).fit(pixels)

# *Replace each pixel value with its corresponding cluster center*

quantized\_image = kmeans.cluster\_centers\_[kmeans.labels\_].reshape(image.shape)

return quantized\_image

# *Example usage*

image = cv2.imread('cameraman.png', cv2.IMREAD\_GRAYSCALE)

n\_clusters = 8 # *Number of gray levels*

quantized\_image = vector\_quantization(image, n\_clusters)

plt.figure(figsize=(10, 5))

plt.subplot(1, 2, 1)

plt.title('Original Image')

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

plt.subplot(1, 2, 2)

plt.title('Quantized Image')

plt.imshow(quantized\_image, cmap='gray')

plt.show()

**Task 4**: Write a python code to perform (a) nearest neighbor (b) Bilinear and (c) Bicubic interpolation of the input image and comment on the observed result.

# *Interpolation functions*

def nearest\_neighbor\_interpolation(image, new\_size):

return cv2.resize(image, new\_size, interpolation=cv2.INTER\_NEAREST)

def bilinear\_interpolation(image, new\_size):

return cv2.resize(image, new\_size, interpolation=cv2.INTER\_LINEAR)

def bicubic\_interpolation(image, new\_size):

return cv2.resize(image, new\_size, interpolation=cv2.INTER\_CUBIC)

# *Example usage*

image = cv2.imread('cameraman.png', cv2.IMREAD\_GRAYSCALE)

new\_size = (512, 512)

nn\_image = nearest\_neighbor\_interpolation(image, new\_size)

bilinear\_image = bilinear\_interpolation(image, new\_size)

bicubic\_image = bicubic\_interpolation(image, new\_size)

plt.figure(figsize=(15, 5))

plt.subplot(1, 4, 1)

plt.title('Original Image')

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

plt.subplot(1, 4, 2)

plt.title('Nearest Neighbor Interpolation')

plt.imshow(nn\_image, cmap='gray')

plt.subplot(1, 4, 3)

plt.title('Bilinear Interpolation')

plt.imshow(bilinear\_image, cmap='gray')

plt.subplot(1, 4, 4)

plt.title('Bicubic Interpolation')

plt.imshow(bicubic\_image, cmap='gray')

plt.show()

**Task 5**: Write a python code to perform resizing of the input image. If the input image size is 256 x 256, first resize it to 128 x 128, then resize the input image to 512 x 512 and comment on the observed result.

# *Example usage*

image = cv2.imread('cameraman.png', cv2.IMREAD\_GRAYSCALE)

# *Resize to 128x128*

resized\_down = cv2.resize(image, (128, 128), interpolation=cv2.INTER\_LINEAR)

# *Resize to 512x512*

resized\_up = cv2.resize(image, (512, 512), interpolation=cv2.INTER\_LINEAR)

plt.figure(figsize=(15, 5))

plt.subplot(1, 3, 1)

plt.title('Original Image')

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

plt.subplot(1, 3, 2)

plt.title('Resized to 128x128')

plt.imshow(resized\_down, cmap='gray')

plt.subplot(1, 3, 3)

plt.title('Resized to 512x512')

plt.imshow(resized\_up, cmap='gray')

plt.show()

**Task 6**: Generate an 8 x 8 image which is given by f[m,n]=|m-n| for m,n=0,1,2,3,4,5,6,7. Pass this image through 3x 3 average filter, 3 x 3 weighted average filter. Use subplot to plot the generated image, average filtered image and weighted average filtered image and comment on the observed result.

# *Generate the 8x8 image*

f = np.abs(np.subtract.outer(np.arange(8), np.arange(8)))

# *Define the 3x3 average filter*

average\_filter = np.ones((3, 3)) / 9

# *Define the 3x3 weighted average filter*

weighted\_average\_filter = np.array([[1, 2, 1],

[2, 4, 2],

[1, 2, 1]]) / 16

# *Apply the filters*

average\_filtered\_image = cv2.filter2D(f, -1, average\_filter)

weighted\_average\_filtered\_image = cv2.filter2D(f, -1, weighted\_average\_filter)

plt.figure(figsize=(15, 5))

plt.subplot(1, 3, 1)

plt.title('Original Image')

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

plt.subplot(1, 3, 2)

plt.title('Average Filtered Image')

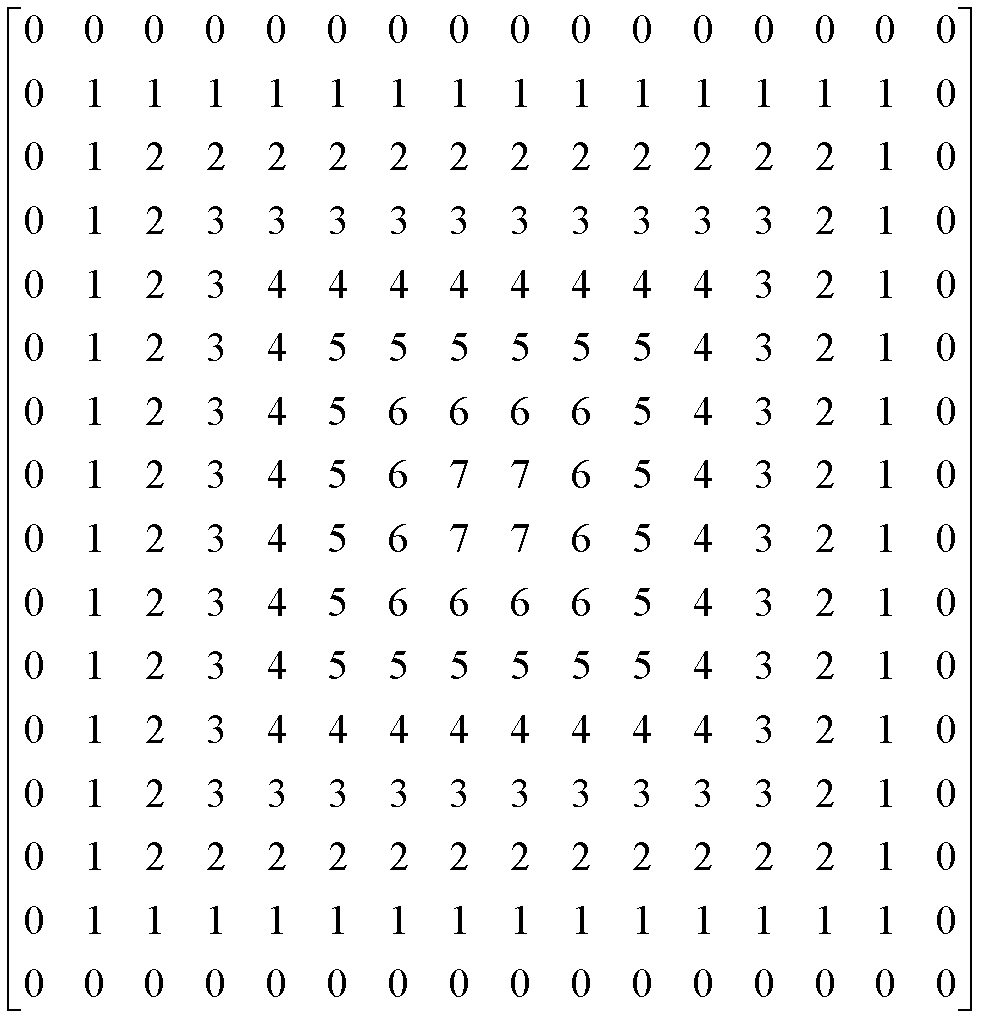
plt.imshow(average\_filtered\_image, cmap='gray')

plt.subplot(1, 3, 3)

plt.title('Weighted Average Filtered Image')

plt.imshow(weighted\_average\_filtered\_image

**Task 7**: Generate the 16 x 16 image which is given below



Use some logic to generate this image. **Hint:** The image has outer ring made of gray level value ‘0’, inner ring of gray level value ‘1’ and so on.

def generate\_ring\_image(size):

image = np.zeros((size, size), dtype=int)

for i in range(size):

for j in range(size):

image[i, j] = min(i, j, size - i - 1, size - j - 1)

return image

image = generate\_ring\_image(16)

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

plt.title("16x16 Ring Image")

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

plt.show()