Name: Bidyut Kr. Das

Id: 221001011060 Batch: BCS-4D

Question 1: Write a program in python to convert colored image to gray image.

Use rations (0.28:0.59:0.10)

Code:

**import** numpy **as** np

**import** matplotlib**.**pyplot **as** plt

**from** PIL **import** Image

**def** rgb\_to\_gray(**image**)**:**

**image** **=** Image**.**open(**image**)

    image\_array **=** np**.**array(**image**)

    r**,** g**,** b **=** image\_array[**:,** **:,** 0]**,** image\_array[**:,** **:,** 1]**,** image\_array[**:,** **:,** 2]

    grayscale **=** 0.28 **\*** r **+** 0.59 **\*** g **+** 0.10 **\*** b

    grayscale\_image **=** grayscale**.**astype(np**.***uint8*)

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

    plt**.**subplot(1**,** 2**,** 1)

    plt**.**title("Original Image")

    plt**.**imshow(image\_array)

    plt**.**axis("off")

    plt**.**subplot(1**,**2**,** 2)

    plt**.**title("Grayscale Image")

    plt**.**imshow(grayscale\_image**,** **cmap=**"gray")

    plt**.**axis("off")

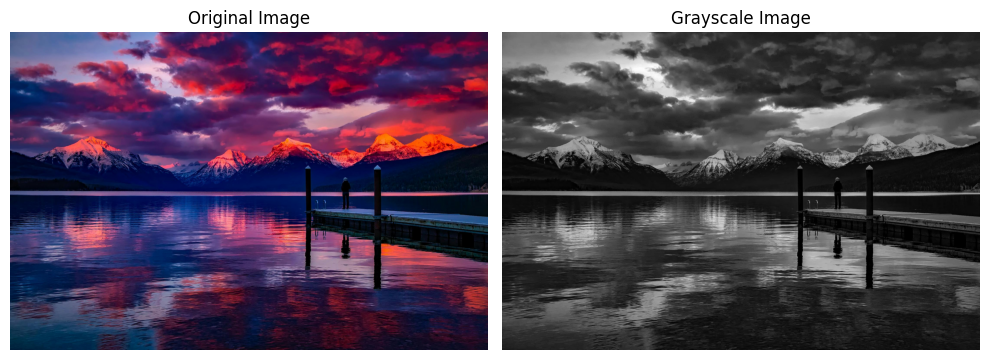
    plt**.**tight\_layout()

    plt**.**show()

**return** grayscale\_image

input\_image **=** './image.jpg'

gray\_image **=** rgb\_to\_gray(input\_image)



Question 2: Given an image perform the following operations accordingly

1. Add salt pepper noise to it
2. Perform mean filter on the noise image
3. Perform gaussian filter on the noise image

Code:

**def** add\_salt\_pepper\_noise(**image,** **salt\_prob,** **pepper\_prob**)**:**

**image** **=** Image**.**open(**image**)

**image** **=** np**.**array(**image**)

    noisy\_image **=** np**.**copy(**image**)

    total\_pixels **=** **image.***size*

    num\_salt **=** **int**(total\_pixels **\*** **salt\_prob**)

    num\_pepper **=** **int**(total\_pixels **\*** **pepper\_prob**)

    coords\_salt **=** [np**.**random**.***randint*(0**,** i**,** num\_salt) **for** i **in** **image.***shape*]

    noisy\_image[coords\_salt[0]**,** coords\_salt[1]] **=** 255  *# Assuming 8-bit grayscale*

    coords\_pepper **=** [np**.**random**.***randint*(0**,** i**,** num\_pepper) **for** i **in** **image.***shape*]

    noisy\_image[coords\_pepper[0]**,** coords\_pepper[1]] **=** 0

**return** noisy\_image

salt\_prob **=** 0.02

pepper\_prob **=** 0.02

salt\_image **=** add\_salt\_pepper\_noise(input\_image**,** salt\_prob**,** pepper\_prob)

image **=** Image**.**open(input\_image)

image\_arr **=** np**.**array(image)

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

plt**.**subplot(1**,**2**,**1)

plt**.**title("Image")

plt**.**imshow(image\_arr)

plt**.**axis("off")

plt**.**subplot(1**,**2**,**2)

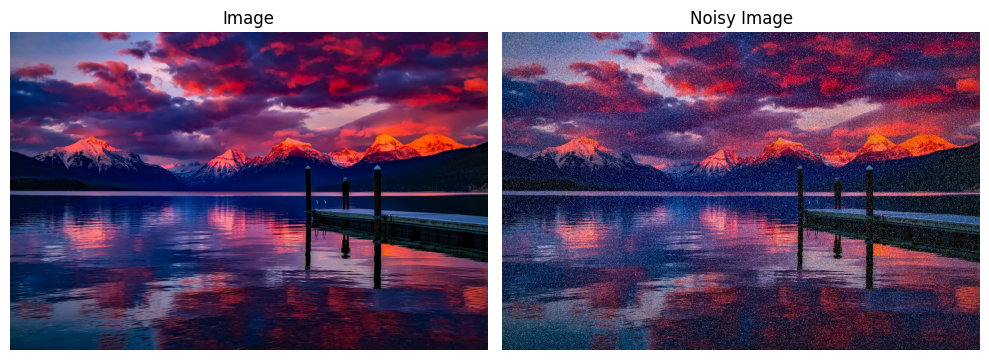
plt**.**title("Noisy Image")

plt**.**imshow(salt\_image)

plt**.**axis("off")

plt**.**tight\_layout()

plt**.**show()



II Code:

**def** mean\_filter(**image,** **kernel\_size** **=** 3)**:**

    height**,** width**,** channels **=** **image.**shape

    pad **=** **kernel\_size** **//** 2

    blurred\_image **=** np**.**zeros\_like(**image,** **dtype=**np**.***uint8*)

**for** c **in** **range**(channels)**:**

        padded\_channel **=** np**.**zeros((height **+** 2 **\*** pad**,** width **+** 2 **\*** pad))

        padded\_channel[pad**:**pad **+** height**,** pad**:**pad **+** width] **=** **image**[**:,** **:,** c]

**for** i **in** **range**(height)**:**

**for** j **in** **range**(width)**:**

                kernel\_region **=** padded\_channel[i**:**i **+** **kernel\_size,** j**:**j **+** **kernel\_size**]

                kernel\_mean **=** np**.**sum(kernel\_region) **/** (**kernel\_size** **\*** **kernel\_size**)

                blurred\_image[i**,** j**,** c] **=** kernel\_mean

**return** blurred\_image

blurred\_image **=** mean\_filter(salt\_image**,** **kernel\_size=**5)

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

plt**.**subplot(1**,**2**,**1)

plt**.**title("Noisy Image")

plt**.**imshow(salt\_image)

plt**.**axis("off")

plt**.**subplot(1**,**2**,**2)

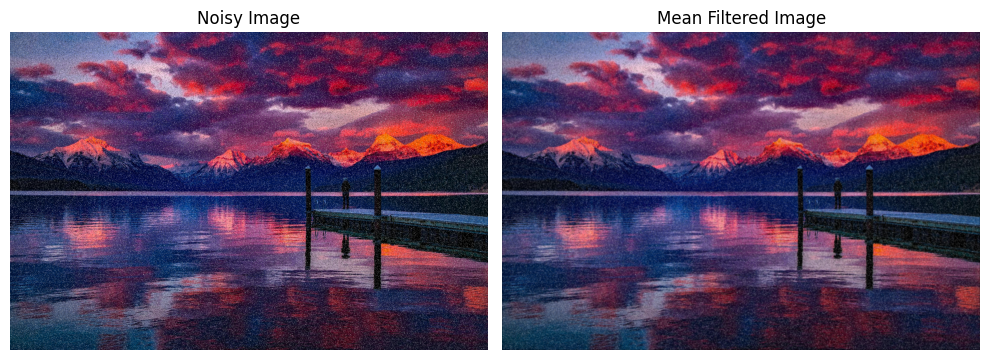
plt**.**title("Mean Filtered Image")

plt**.**imshow(blurred\_image)

plt**.**axis("off")

plt**.**tight\_layout()

plt**.**show()



III Code:

*#creation of gaussian\_kernel*

**def** gaussian\_kernel(**size,** **sigma=**1)**:**

    kernel\_1d **=** np**.**linspace(**-size//**2**,** **size//**2**,** **size**)

    kernel\_1d **=** np**.***exp*(**-**0.5 **\*** (kernel\_1d **/** **sigma**) **\*\*** 2

*# Normalize the kernel so that the sum of all values equals 1*

    kernel\_1d **/=** np**.**sum(kernel\_1d)

*# Create a 2D kernel by taking the outer product of the 1D kernel with itself*

    kernel\_2d **=** np**.**outer(kernel\_1d**,** kernel\_1d)

*# Normalize the 2D kernel (this is just a safeguard, should already sum to 1)*

    kernel\_2d **/=** np**.**sum(kernel\_2d)

**return** kernel\_2d

**def** gaussian\_blur\_color(**image,** **kernel\_size=**5**,** **sigma=**1)**:**

**if** len(**image.**shape) **!=** 3 **or** **image.**shape[2] **not** **in** [3**,** 4]**:**  *# Check for RGB or RGBA*

**raise** **ValueError**("Input image must be a color image with 3 or 4 channels (e.g., RGB or RGBA).")

    kernel **=** gaussian\_kernel(**kernel\_size,** **sigma**)

    height**,** width**,** channels **=** **image.**shape

    pad **=** **kernel\_size** **//** 2

    blurred\_image **=** np**.**zeros\_like(**image,** **dtype=**np**.***uint8*)

**for** c **in** **range**(channels)**:**

        channel **=** **image**[**:,** **:,** c]

        padded\_channel **=** np**.**pad(channel**,** ((pad**,** pad)**,** (pad**,** pad))**,** **mode=**'constant'**,** **constant\_values=**0)

**for** i **in** **range**(height)**:**

**for** j **in** **range**(width)**:**

                region **=** padded\_channel[i**:**i **+** **kernel\_size,** j**:**j **+** **kernel\_size**]

                blurred\_image[i**,** j**,** c] **=** np**.**sum(region **\*** kernel)

**return** blurred\_image

kernel\_size **=** 5

sigma **=** 1

blurred\_image **=** gaussian\_blur\_color(salt\_image**,** kernel\_size**,** sigma)

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

plt**.**subplot(1**,**2**,**1)

plt**.**title("Noisy Image")

plt**.**imshow(salt\_image**,** **cmap=**"gray")

plt**.**axis("off")

plt**.**subplot(1**,**2**,**2)

plt**.**title("Gaussian Filter")

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

plt**.**axis("off")

plt**.**tight\_layout()

plt**.**show()

