**ADVANCE DIGITAL IMAGE PROCESSING**

**PYTHON PROGRAMMING FOR IMAGE UNDERSTANDING AND ENHANCEMENT**

**SHUBHASH S  
25MES1015**

**COLOR-PLANE INFORMATION**

**CODE:**

import numpy as np

import matplotlib.pyplot as plt

import matplotlib.image as mpimg

img=mpimg.imread('peacock.jpg')

plt.figure(1)

plt.imshow(img)

plt.title('Original Image')

imgR=img[:,:,0]

imgG=img[:,:,1]

imgB=img[:,:,2]

plt.figure(2)

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

plt.title('Red Channel')

plt.figure(3)

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

plt.title('Green Channel')

plt.figure(4)

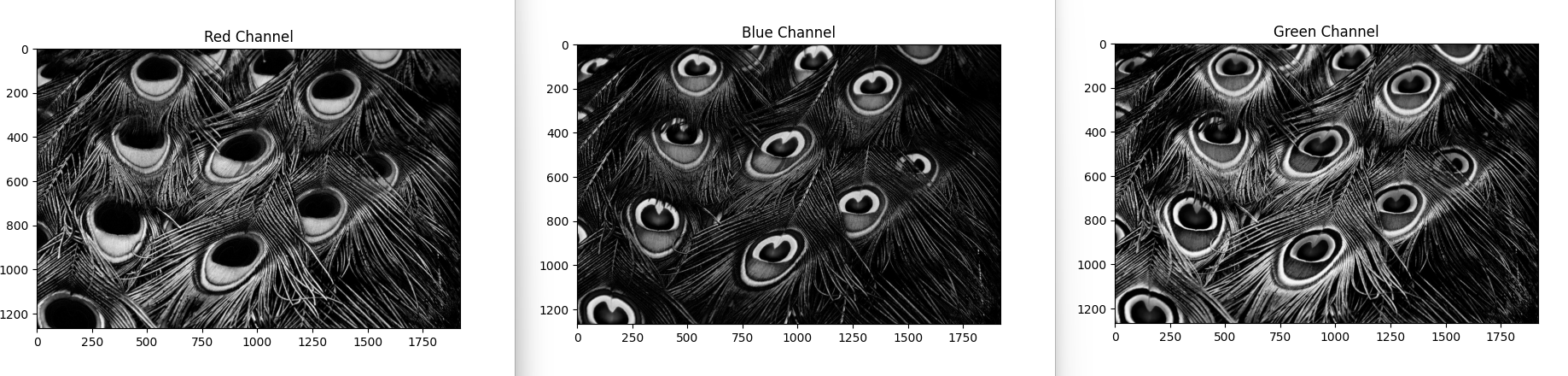
plt.imshow(imgB,cmap='gray')

plt.title('Blue Channel')

plt.show()

**OUTPUT:**





**INFERENCE:**

Each image shows a single RGB channel converted to grayscale.  
The brighter the channel at a particular location, the closer that region is to white; lower intensity regions appear closer to black.

**HISTOGRAM**

**CODE:**

import numpy as np

import matplotlib.pyplot as plt

import matplotlib.image as mpimg

img=mpimg.imread('low\_contrast.jpg')

def rgb2gray(rgb):

    r, g, b = rgb[:,:,0], rgb[:,:,1], rgb[:,:,2]

    gray = 0.2989 \* r + 0.5870 \* g + 0.1140 \* b

    return gray

img\_gray=rgb2gray(img)

plt.figure(1)

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

plt.figure(2)

#plt.show()

#plt.hist(img\_gray)

plt.hist(img\_gray.ravel(), bins=256, range=(0, 256))

plt.title("Histogram of given image")

plt.xlabel("Pixel values")

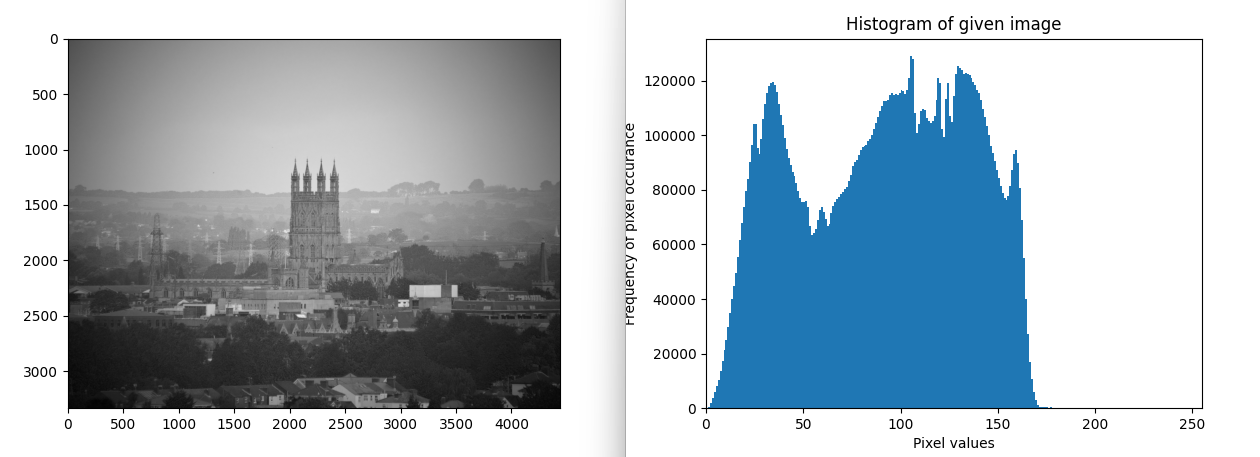
plt.ylabel("Frequency of pixel occurance")

plt.xlim([0, 255])

#plt.imshow(img\_gray,cmap='gray')

plt.show()

**OUTPUT:**



**INFERENCE:**

The histogram plot clearly shows the intensity levels of the given image. Most of the intensity values lie between 0 and around 180, and beyond that range there are very few or no pixels.

**LOG TRANSFORMATION**

**CODE**

import numpy as np

import matplotlib.pyplot as plt

import cv2

img = cv2.imread('child.jpg')

gray = cv2.cvtColor(img, cv2.COLOR\_BGR2GRAY)

gray\_float = gray.astype(np.float32)

c = 255 / np.log(1 + np.max(gray\_float))

log\_img = c \* np.log(1 + gray\_float)

log\_img = np.uint8(log\_img)

# Display Original Image

plt.figure(1)

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

plt.title("Original Grayscale Image")

plt.axis('off')

# Histogram of Original Image

plt.figure(2)

plt.hist(gray.ravel(), bins=256)

plt.title("Histogram of Original Image")

# Display Log Transformed Image

plt.figure(3)

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

plt.title("Log Transformed Image")

plt.axis('off')

# Histogram of Log Transformed Image

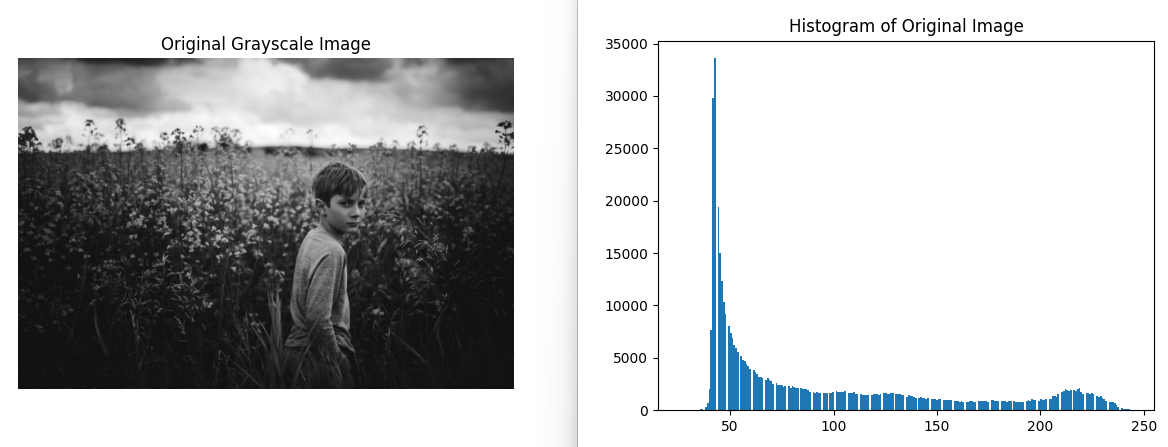
plt.figure(4)

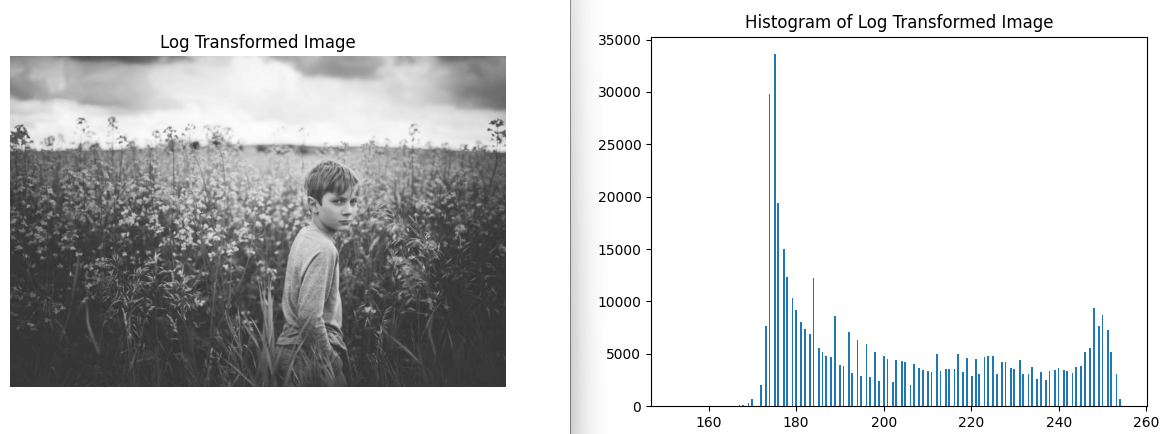
plt.hist(log\_img.ravel(), bins=256)

plt.title("Histogram of Log Transformed Image")

plt.show()

**OUTPUT**

****

****

**INFERENCE**

It is observed that the darker regions of the image are enhanced by expanding low-intensity pixel values while compressing higher-intensity values, resulting in improved visibility of details in dark areas and a more balanced intensity distribution in the histogram.

**GAMMA TRANSFORMATION**

**CODE**

import numpy as np

import matplotlib.pyplot as plt

import matplotlib.image as mpimg

import cv2

img = cv2.imread('peacock.jpg')

def rgb2gray(rgb):

    r, g, b = rgb[:,:,0], rgb[:,:,1], rgb[:,:,2]

    gray = 0.2989 \* r + 0.5870 \* g + 0.1140 \* b

    return gray

# Trying 4 gamma values.

a=rgb2gray(img)

plt.figure()

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

plt.title("Original image")

for gamma in [0.1, 0.5, 1.2, 2.2]:

    gamma\_corrected = np.array(255\*(a / 255) \*\* gamma, dtype = 'uint8')

    print('gamma\_transformed '+str(gamma)+'.jpg')

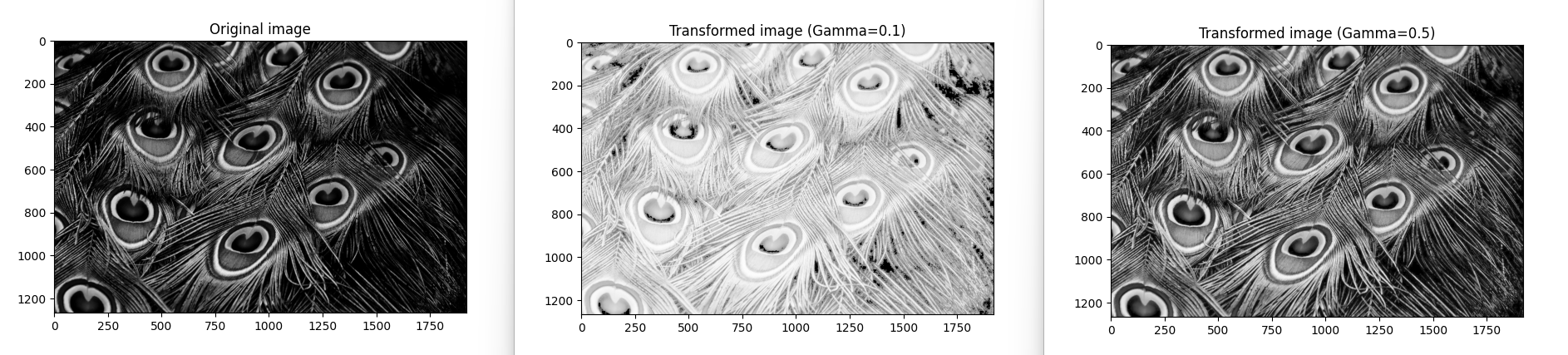
    plt.figure()

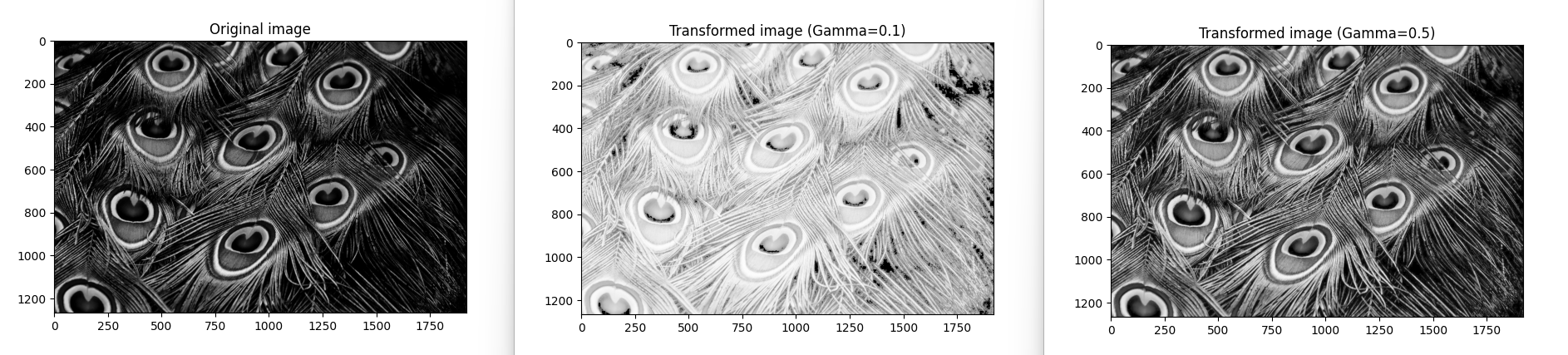
    plt.imshow(gamma\_corrected, cmap='gray')

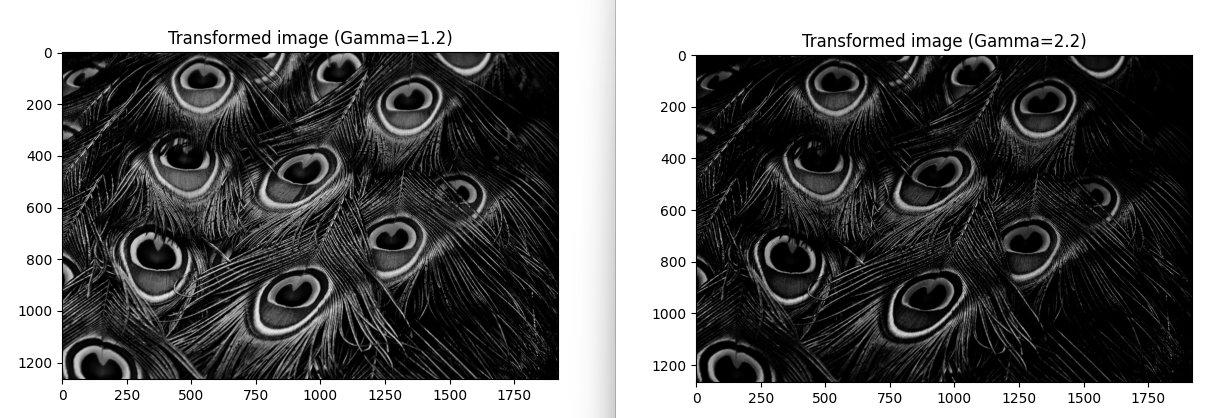
    plt.title(f"Transformed image (Gamma={gamma})")

plt.show()

**OUTPUT**

****

****

****

**INFERENCE**

From the results, it is observed that when the gamma value is less than 1, the image becomes brighter and the contrast increases. As the gamma value increases beyond 1, the image becomes darker.

**GAUSSIAN AND MEDIAN FILTERS**

**CODE**

import numpy as np

import matplotlib.pyplot as plt

import matplotlib.image as mpimg

from scipy import ndimage # Multi-dimensional image processing

img=mpimg.imread('noice.jpg')

def rgb2gray(rgb):

    r, g, b = rgb[:,:,0], rgb[:,:,1], rgb[:,:,2]

    gray = 0.2989 \* r + 0.5870 \* g + 0.1140 \* b

    return gray

img=rgb2gray(img)

img\_noise = img + 0.05\* np.random.randn(\*img.shape)

blurred\_img1=ndimage.gaussian\_filter(img\_noise, sigma=1.5) # Multidimensional Gaussian filter

blurred\_img2=ndimage.median\_filter(img\_noise,5) # Calculate a multidimensional median filter

plt.figure(1)

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

plt.title('Image with Noise')

plt.figure(2)

plt.imshow(blurred\_img1, cmap='gray')

plt.title('Gaussian Filter (Blur)')

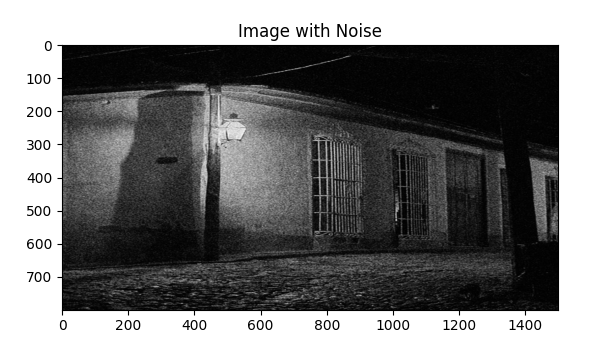
plt.figure(3)

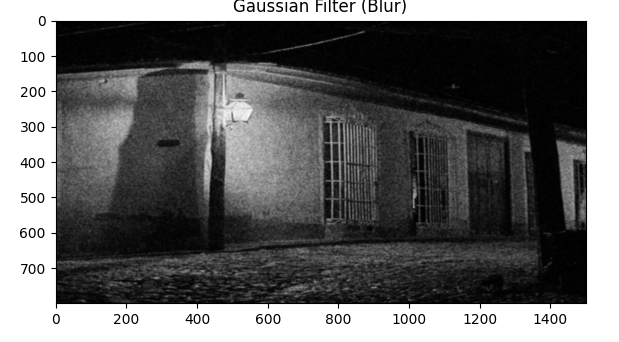
plt.imshow(blurred\_img2, cmap='gray')

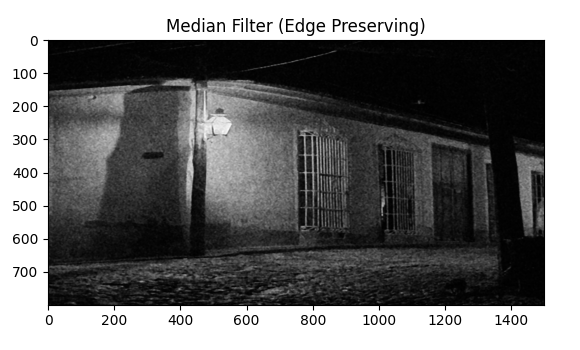
plt.title('Median Filter (Edge Preserving)')

plt.show()

**OUTPUT**







**INFERENCE**

The Gaussian filter blurs the noisy image, which helps remove noise, but it also reduces image quality by blurring details. In contrast, the median filter reduces noise while preserving edges, so the edges are not blurred.