01.

import cv2 as cv

import numpy as np

import matplotlib.pyplot as plt

img = cv.imread('emma\_gray.jpg', cv.IMREAD\_GRAYSCALE) #importing the image

#remapping the input intensity values between 50 to 150

def pixVal(val):

    if (50 <= val and val <= 150): #define input range

        return (155/100) \* (val - 50) + 100 #remapping equation

    else:

        return val

#verctorizing the pixel values as an numpy array

pixVal\_vec = np.vectorize(pixVal)

#indexing the transformartion pixVal\_vec with the values of pixels in the img

img\_new = pixVal\_vec(img)

#show output

plt.subplot(121),plt.imshow(img, cmap='gray', vmin=0, vmax=255)

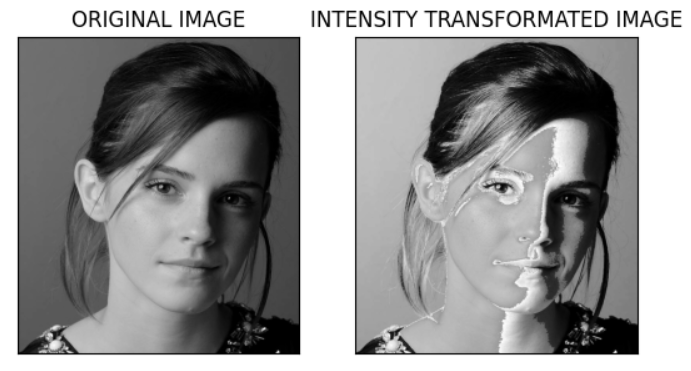
plt.title('ORIGINAL IMAGE'), plt.xticks([]), plt.yticks([])

plt.subplot(122),plt.imshow(img\_new, cmap='gray', vmin=0, vmax=255)

plt.title('INTENSITY TRANSFORMATED IMAGE'), plt.xticks([]), plt.yticks([])

plt.show()

OUTPUT:



02.

03.

import cv2 as cv

import numpy as np

import matplotlib.pyplot as plt

im1 = cv.imread('highlights\_and\_shadows.jpg', cv.IMREAD\_COLOR) #read image

im1 = cv.cvtColor(im1, cv.COLOR\_BGR2RGB) #convert BGR to RGB

im\_L = cv.cvtColor(im1, cv.COLOR\_RGB2LAB) #convert RGB to L\*a\*b color space

gamma = 2 #lightness (L plane) gamma value

arr = np.array([(i/255)\*\*gamma\*255 for i in range (0, 256)]).astype(np.uint8) #define gamma array

im\_gamma = arr[im\_L] #indexing the transformartion arr with the values of pixels in the im\_L

#show output

plt.subplot(121),plt.imshow(im1, vmin=0, vmax=255)

plt.title('ORIGINAL IMAGE'), plt.xticks([]), plt.yticks([])

plt.subplot(122),plt.imshow(im\_L, vmin=0, vmax=255)

plt.title('GAMMA CORRECTED Lab IMAGE'), plt.xticks([]), plt.yticks([])

plt.show()

#show histograms original vs corrected image

hist\_r = cv.calcHist([im\_gamma], [0], None, [256], [0,256])

hist\_g = cv.calcHist([im\_gamma], [1], None, [256], [0,256])

hist\_b = cv.calcHist([im\_gamma], [2], None, [256], [0,256])

hist\_r\_g = cv.calcHist([im1], [0], None, [256], [0,256])

hist\_g\_g = cv.calcHist([im1], [1], None, [256], [0,256])

hist\_b\_g = cv.calcHist([im1], [2], None, [256], [0,256])

plt.subplot(121), plt.plot(hist\_r, 'r', hist\_g, 'g', hist\_b, 'b')

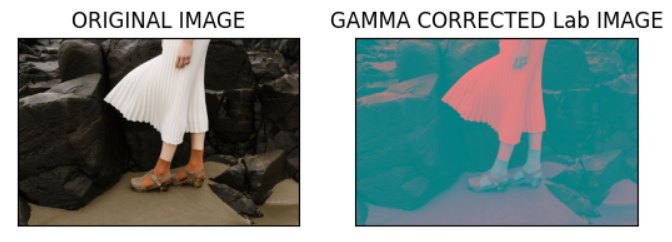
plt.title('ORIGINAL IMAGE'), plt.xlim([0,256])

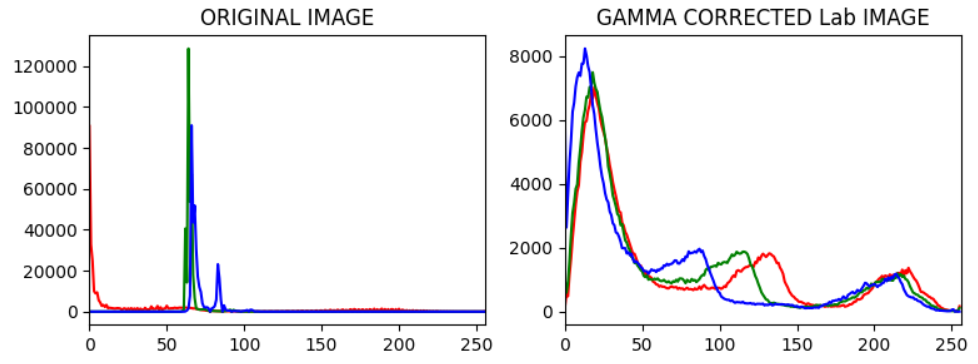
plt.subplot(122), plt.plot(hist\_r\_g, 'r', hist\_g\_g, 'g', hist\_b\_g, 'b')

plt.title('GAMMA CORRECTED Lab IMAGE'), plt.xlim([0,256])

plt.show()

OUTPUT:





04.

import cv2 as cv

import numpy as np

import matplotlib.pyplot as plt

im = cv.imread('shells.png', cv.IMREAD\_GRAYSCALE) #import image

hist = cv.calcHist([im], [0], None, [256], [0,256]) #calculate histogram (intensity distribution of an image)

equ = cv.equalizeHist(im) #histogram equlization (improves the contrast in an image)

hist\_aft = cv.calcHist([equ], [0], None, [256], [0,256]) #calculate histo. of equi. one

#show output

fig, ax = plt.subplots(1,2, figsize=(8,16))

ax[0].imshow(im, cmap='gray', vmin=0, vmax=255)

ax[0].xaxis.tick\_top()

ax[1].imshow(equ, cmap='gray', vmin=0, vmax=255)

ax[1].xaxis.tick\_top()

plt.show()

#plot histograms

plt.figure(figsize=(9, 3))

plt.subplot(121), plt.plot(hist)

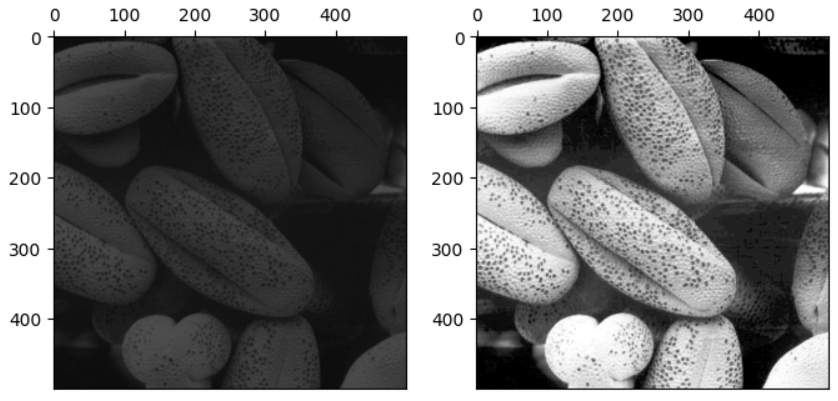
plt.title('ORIGINAL IMAGE HISTOGRAM'), plt.xlim([0,256])

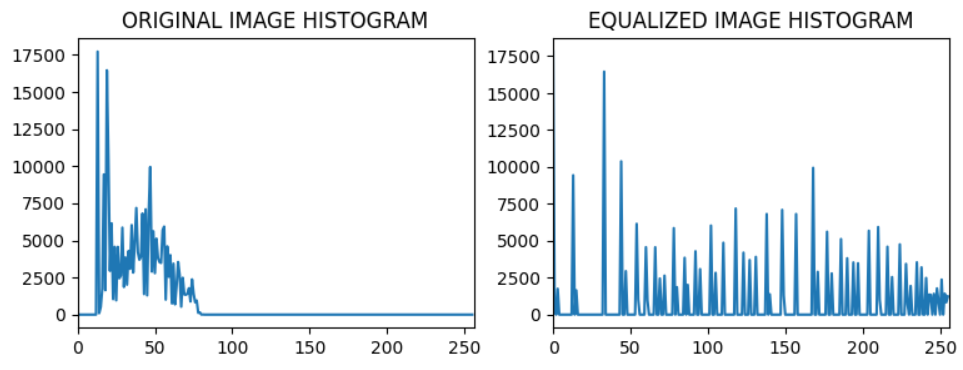
plt.subplot(122), plt.plot(hist\_aft)

plt.title('EQUALIZED IMAGE HISTOGRAM'), plt.xlim([0,256])

plt.show()

OUTPUT:





05.

import cv2

from matplotlib import pyplot as plt

import numpy as np

im\_small = cv2.imread('im01small.png', cv.IMREAD\_COLOR) #read small image

im\_small = cv.cvtColor(im\_small, cv.COLOR\_BGR2RGB)

im\_large = cv2.imread('im01.png', cv.IMREAD\_COLOR) #read large image

im\_large = cv.cvtColor(im\_large, cv.COLOR\_BGR2RGB)

#zoom the image by giving factor s = 4

s = 4

zoom\_NN = cv2.resize(im\_small, None, interpolation=cv2.INTER\_NEAREST , fx=s, fy=s) #nearest-neighbor

zoom\_BI = cv2.resize(im\_small, None, interpolation=cv2.INTER\_LINEAR , fx=s, fy=s) #bilinear interpolation.

#show output

fig, ax = plt.subplots(1,3, figsize=(20,60))

ax[0].imshow(im\_small, vmin=0, vmax=255)

ax[0].xaxis.tick\_top()

ax[0].set\_title("Small Image")

ax[1].imshow(zoom\_NN, vmin=0, vmax=255)

ax[1].xaxis.tick\_top()

ax[1].set\_title("Zoom Image - Nearest-Neighbor")

ax[2].imshow(zoom\_BI, vmin=0, vmax=255)

ax[2].xaxis.tick\_top()

ax[2].set\_title("Zoom Image - Bilinear Interpolation.")

plt.show()

#computing SSD

SSD =  np.sum((np.array(zoom\_NN, dtype=np.float32) - np.array(im\_large, dtype=np.float32))\*\*2)

SSD

OUTPUT: