**Assignment – 1**

1. Write a program on Histogram Equalization for global enhancement. A plot of the histogram before and after the equalization should also be shown. You may Run your algorithm on MATLAB/PYTHON/C and print out the histogram on the screen, or plot it

**Justification**: The Image I have chosen is moon’s surface image and used histogram equalization to enhance the visible features in the image, for color image equalization I took an image with low brightness to improve it’s visibility.

**Code:**

For monochrome:

import cv2

import numpy as np

import matplotlib.pyplot as plt

from graph\_helper import plot\_graph

def myEqualizeHist(img):

    hist,bins=np.histogram(img.flatten(),256,[0,256])

    cdf=np.cumsum(hist)

    cdf\_m=np.ma.masked\_equal(cdf,0)

    cdf\_m=(cdf\_m-cdf\_m.min())\*255/(cdf\_m.max()-cdf\_m.min())

    cdf\_final=np.ma.filled(cdf\_m,0).astype('uint8')

    img=cdf\_final[img]

    return img

def main(path,function):

    img=cv2.imread(path,0)

    plot\_graph.display\_histogram(img,'Image before Equalization')

    equalized\_img=function(img)

    plot\_graph.display\_histogram(equalized\_img,'Image after Equalization')

    comp\_stack=np.hstack((img,equalized\_img))

    cv2.imshow('Image before and after histigram equilization',comp\_stack)

    cv2.waitKey(0)

    cv2.destroyAllWindows()

if \_\_name\_\_=='\_\_main\_\_':

    print('Predefined Histogram Equalizer')

    main('./images/moon-crater.jpeg',cv2.equalizeHist)

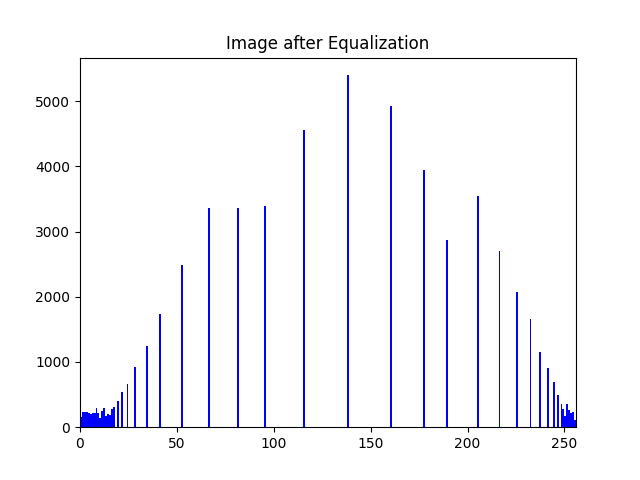
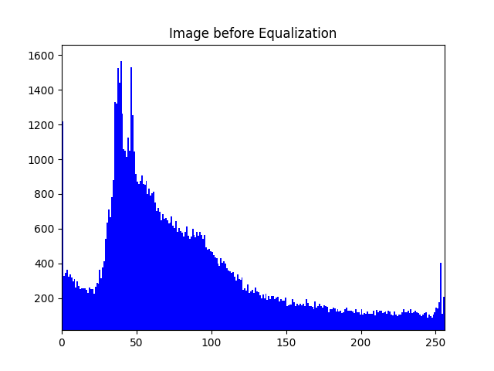
    print('My Histogram Equalizer')

    main('./images/moon-crater.jpeg',myEqualizeHist)

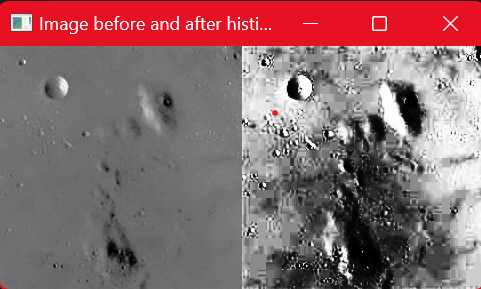
**Outputs:**

Predefined:

Histograms:

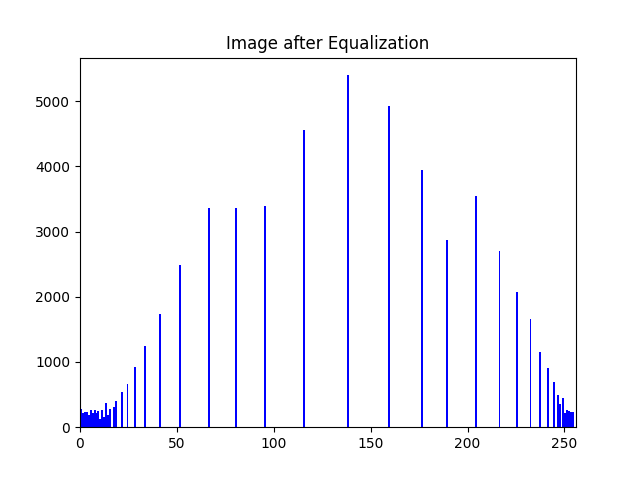
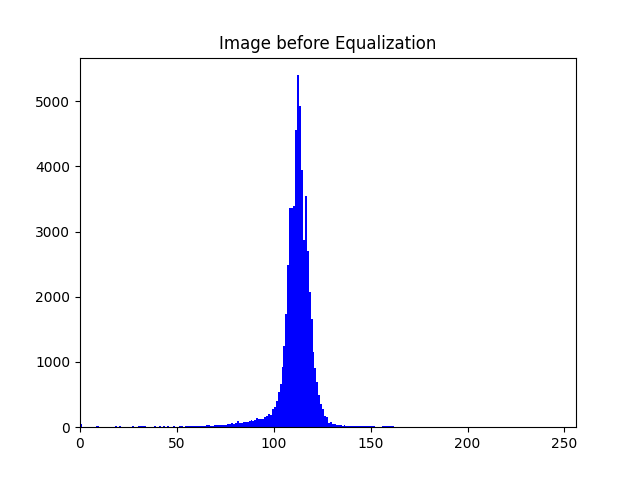


Images:

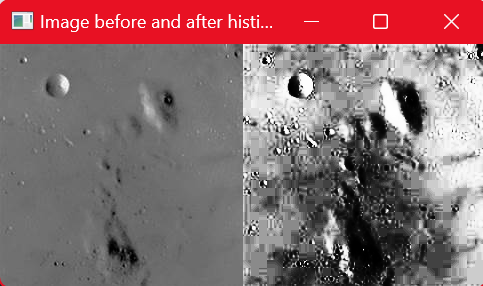


User Defined:

Histogram



Images:



**For Color**:

import cv2

import numpy as np

from PIL import Image

from graph\_helper import plot\_graph

def hist\_equalizer(color):

    h,bin\_b=np.histogram(color.flatten(),256,[0,256])

    cdf=np.cumsum(h)

    cdf\_m=np.ma.masked\_equal(cdf,0)

    cdf\_m=(cdf\_m - cdf\_m.min())\*255/(cdf\_m.max()-cdf\_m.min())

    cdf\_final=np.ma.filled(cdf\_m,0).astype('uint8')

    return cdf\_final[color]

def myhistogram\_equalization(img):

    blue,green,red=cv2.split(img)

    equ\_b=hist\_equalizer(blue)

    equ\_g=hist\_equalizer(green)

    equ\_r=hist\_equalizer(red)

    equ=cv2.merge((equ\_b,equ\_g,equ\_r))

    return equ

def histogram\_equalization(img):

    blue,green,red=cv2.split(img)

    equ\_b=cv2.equalizeHist(blue)

    equ\_g=cv2.equalizeHist(green)

    equ\_r=cv2.equalizeHist(red)

    equ=cv2.merge((equ\_b,equ\_g,equ\_r))

    return equ

if \_\_name\_\_=='\_\_main\_\_':

    orginal=cv2.imread('./images/moon-crater-colors.jpeg')

    plot\_graph.display\_histogram(orginal,'Histogram before Equalization')

    #final=histogram\_equalization(orginal)

    final=myhistogram\_equalization(orginal)

    plot\_graph.display\_histogram(final,'Histogram after Equalization')

    stacked = np.hstack((orginal,final))

    cv2.imshow('Before and After',stacked)

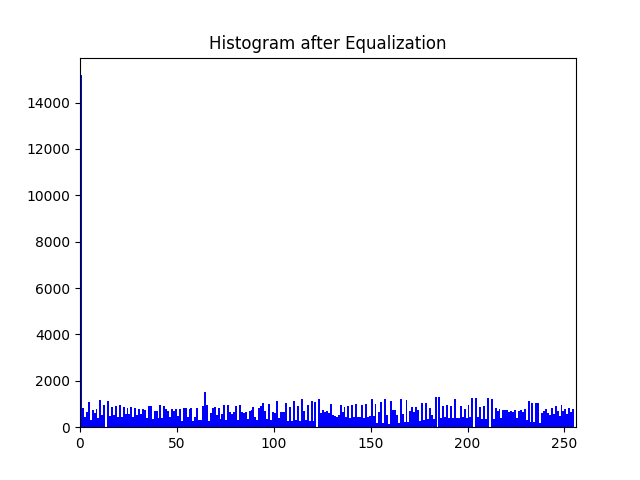
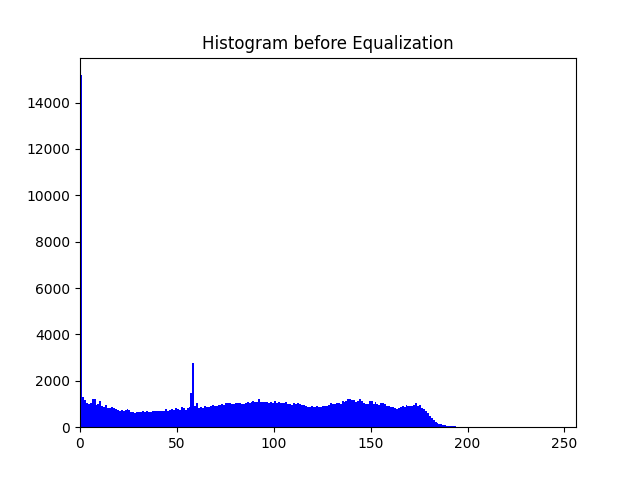
    cv2.waitKey(0)

    cv2.destroyAllWindows()

**Outputs**:

Predefined:

Histograms:



Images:



1. Convert rgb2gray to convert it from RGB to Monochrome

**Code:**

import cv2

import numpy as np

import PIL.Image as im

def convert\_to\_grayscale(image\_path):

    img=cv2.imread(image\_path)

    B,G,R=cv2.split(img)

    gray=0.299\*R + 0.587\*G + 0.114\*B

    return im.fromarray(gray)

if \_\_name\_\_ == "\_\_main\_\_":

    grayscale\_image=convert\_to\_grayscale('./images/black-hole.jpeg')

    grayscale\_image.show()

**Output:**

**** 

1. Make use of RGB2HSI and experiment it in HIS color model

**Code**:

import cv2

from PIL import Image

from matplotlib import pyplot as plt

import numpy as np

from graph\_helper import plot\_graph

def buildin\_rgb\_hsi(path):

    image=cv2.imread(path)

    plot\_graph.display\_histogram(image,'Histogram of RGB image')

    hsv\_image=cv2.cvtColor(image, cv2.COLOR\_BGR2HSV)

    plot\_graph.display\_histogram(hsv\_image,'Histogram of HSV image')

    Image.fromarray(hsv\_image).show()

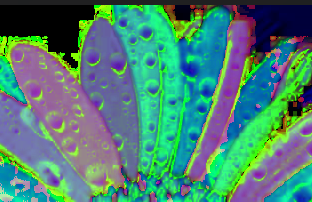
    cv2.waitKey(0)

    cv2.destroyAllWindows()

if \_\_name\_\_=='\_\_main\_\_':

    buildin\_rgb\_hsi('./images/colors.jpeg')

**Output**:



1. Make it poor contrast:
2. Dark Image (Subtract/Gamma correction), Bright Image (Add/Gamma Correction)

**Code**:

import numpy as np

import cv2

from matplotlib import pyplot as plt

import numpy as np

from graph\_helper import plot\_graph

def adjust\_gamma(image,gamma=1.0):

    img\_normalized = img/255.0

    corrected\_img = np.power(img\_normalized, gamma)

    corrected\_img = np.uint8(corrected\_img \* 255)

    return corrected\_img

def plot\_gamma\_transformation(image,gamma=1.0):

    adjusted\_image=adjust\_gamma(image,gamma)

    original\_pixels=image.flatten()

    adjusted\_pixels=adjusted\_image.flatten()

    plt.figure(figsize=(8,6))

    plt.scatter(original\_pixels,adjusted\_pixels,alpha=0.1,s=1)

    plt.plot([0,255],[0,255],color='red',linestyle='--')

    plt.title(f'Gamma Correction (Gamma={gamma})')

    plt.xlabel('Original Pixel Value')

    plt.ylabel('Adjusted Pixel Value')

    plt.xlim([0,255])

    plt.ylim([0,255])

    plt.grid(True)

    plt.show()

if \_\_name\_\_=='\_\_main\_\_':

    path=input("Enter the path of image: ")

    gamma=float(input("Enter the gamma falue \n>1 - Darken\n<1 - lighten\n1-same image\n"))

    img=cv2.imread(path)

    plot\_graph.display\_histogram(img,'Histogram before Adjusting brightness')

    gamma\_corrected=adjust\_gamma(img,gamma)

    plot\_graph.display\_histogram(gamma\_corrected,f'Histogram after Adjusting brightness gamma={gamma}')

    stacked=np.hstack((img,gamma\_corrected))

    cv2.imshow('Orginal and Gamma corrected image',stacked)

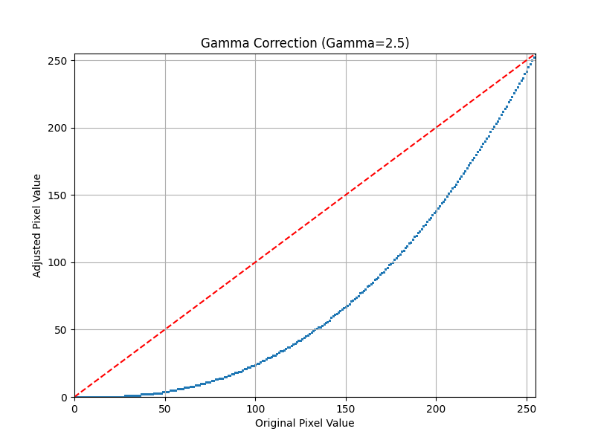
    cv2.waitKey(0)

    cv2.destroyAllWindows()

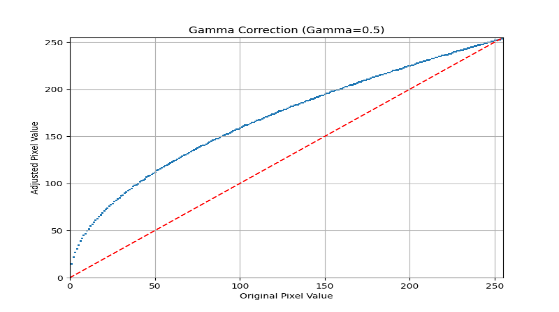
    plot\_gamma\_transformation(img,gamma)

**Output**:

Brighter Image: -(Gamma 2.5)



Lighter Image: - (Gamma 0.5)

1. Low contrast (Thresholding) , High Contrast (a high contrast image usually exhibits a bimodal histogram with clear separation between the two predominant modes)

**Code**:

import numpy as np

from graph\_helper import plot\_graph

import cv2

import numpy as np

def low\_contrast(path):

    image=cv2.imread(path,0)

    plot\_graph.display\_histogram(image,'Histogram before applying Threshold')

    \_,thresholded\_image=cv2.threshold(image,200,255,cv2.THRESH\_BINARY)

    plot\_graph.display\_histogram(image,'Histogram after applying Threshold')

    stack=np.hstack((image,thresholded\_image))

    cv2.imshow('Comparision',stack)

    cv2.waitKey(0)

    cv2.destroyAllWindows()

def high\_contrast(path):

    image=cv2.imread(path,0)

    plot\_graph.display\_histogram(image,'Histogram before CLAHE')

    clahe=cv2.createCLAHE(clipLimit=2.0,tileGridSize=(8,8))

    hc\_image=clahe.apply(image)

    plot\_graph.display\_histogram(hc\_image,'Histogram after CLAHE')

    stack=np.hstack((image,hc\_image))

    cv2.imshow('Comparision',stack)

    cv2.waitKey(0)

    cv2.destroyAllWindows()

if \_\_name\_\_=='\_\_main\_\_':

    path=input('Enter the path: ')

    low\_contrast(path)

    high\_contrast(path)

**Output**:

Low Contrast: (Thresholding)



High Contrast: (CLAHE)



1. Write a program on Log transformation, Gamma correction, image negative and contrast stretching. You may Run your algorithm on MATLAB/PYTHON and print out the histogram on the screen, or plot it. A.
   1. **Log Transformation**

**Code**:

import cv2

import numpy as np

from matplotlib import pyplot as plt

def log\_transform(img):

    c=255/np.log(1+np.max(img))

    log\_image=c\*(np.log(img + 1))

    log\_image=np.array(log\_image, dtype=np.uint8)

    return log\_image

def plot\_log\_transform(img):

    adjusted\_image=log\_transform(img)

    original\_pixels=img.flatten()

    adjusted\_pixels=adjusted\_image.flatten()

    plt.figure(figsize=(8,6))

    plt.scatter(original\_pixels,adjusted\_pixels,alpha=0.1,s=1)

    plt.plot([0,255],[0,255],color='red',linestyle='--')

    plt.title(f'Log Transformation')

    plt.xlabel('Original Pixel Value')

    plt.ylabel('Adjusted Pixel Value')

    plt.xlim([0,255])

    plt.ylim([0,255])

    plt.grid(True)

    plt.show()

if \_\_name\_\_=='\_\_main\_\_':

    img=cv2.imread('./images/log-transform.png')

    log\_img=log\_transform(img)

    stack=np.hstack((img,log\_img))

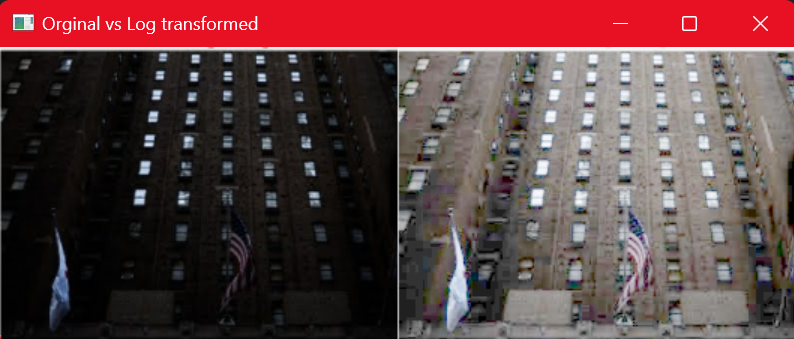
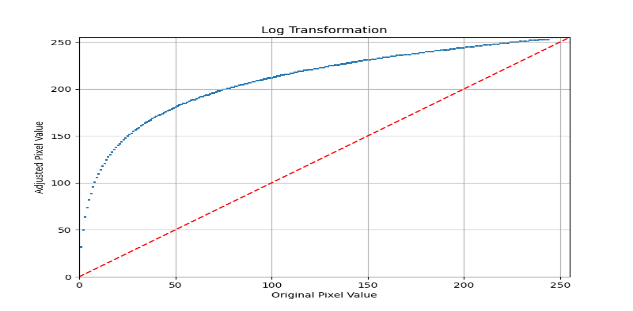
    cv2.imshow('Orginal vs Log transformed',stack)

    cv2.waitKey(0)

    cv2.destroyAllWindows

    plot\_log\_transform(img)

**Output**:



* 1. Gamma Correction

**Code**

Same as brightness adjustment code in 4a

**Output**

Histogram:

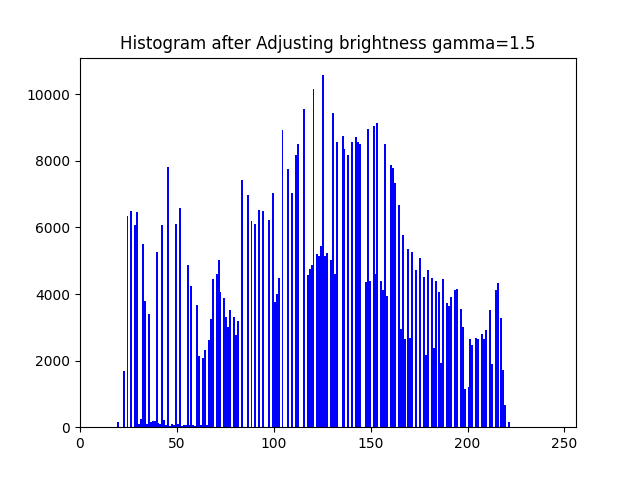
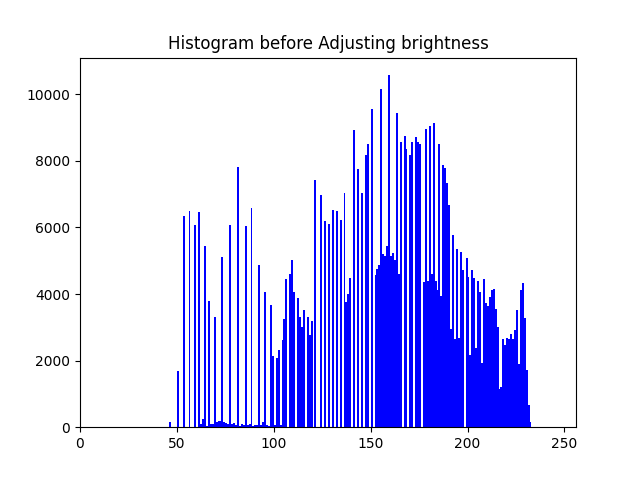
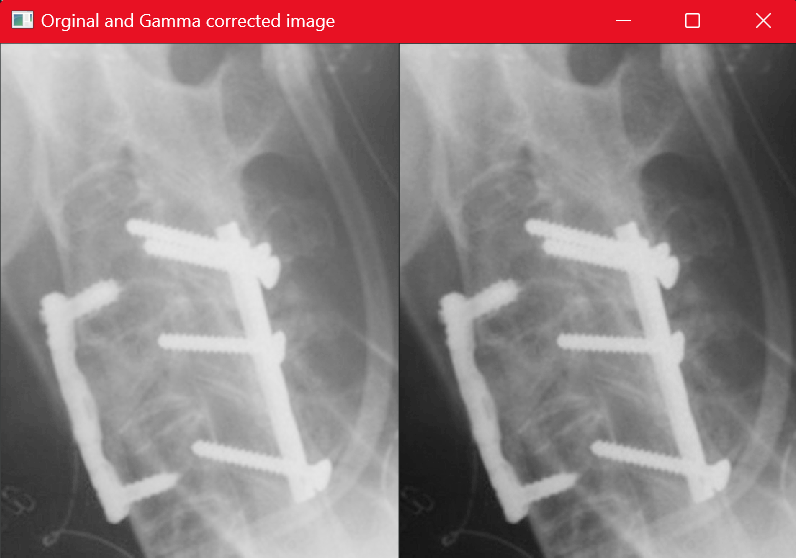


Image:



* 1. Image Negative

**Code**:

import cv2

gray=cv2.imread('./images/colors.jpeg',0)

gray\_neg=cv2.bitwise\_not(gray)

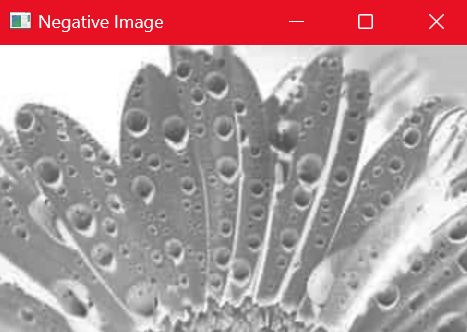
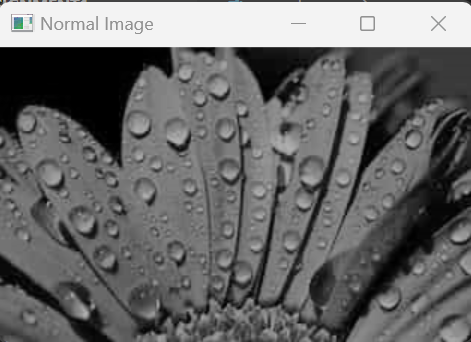
cv2.imshow('Normal Image',gray)

cv2.imshow('Negative Image',gray\_neg)

cv2.waitKey(0)

cv2.destroyAllWindows()

**Output**:



* 1. **Contrast Stretching**

**Code:**

import cv2

import numpy as np

def contrast\_stretch(img):

    min\_val=np.min(img)

    max\_val=np.max(img)

    stretched\_img=(img-min\_val)\*(255.0/(max\_val-min\_val))

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

    return stretched\_img

def contrast\_stretch\_color(img):

    channels=cv2.split(img)

    stretched\_channels=[]

    for channel in channels:

        min\_val=np.min(channel)

        max\_val=np.max(channel)

        stretched\_channel=(channel-min\_val)\*(255.0/(max\_val-min\_val))

        stretched\_channel=np.uint8(stretched\_channel)

        stretched\_channels.append(stretched\_channel)

    stretched\_img=cv2.merge(stretched\_channels)

    return stretched\_img

def main(path,key=0):

    img=cv2.imread(path,key)

    stretched\_img=contrast\_stretch(img)

    stack=np.hstack((img,stretched\_img))

    cv2.imshow('Original vs Contrast Stretched',stack)

    cv2.waitKey(0)

    cv2.destroyAllWindows()

if \_\_name\_\_=='\_\_main\_\_':

    main('images\low-contrast.jpeg',0)

    main('images\low-contrast.jpeg',3)

**Output:**

Black & White:



Color:

