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:

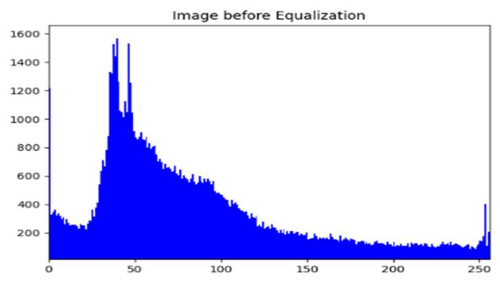
   For monochrome:

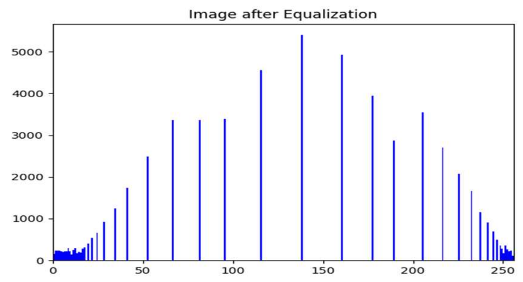
```python
import cv2
import numpy as np
import matplotlib.pyplot as plt
from graph_helper import plot_graph

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:

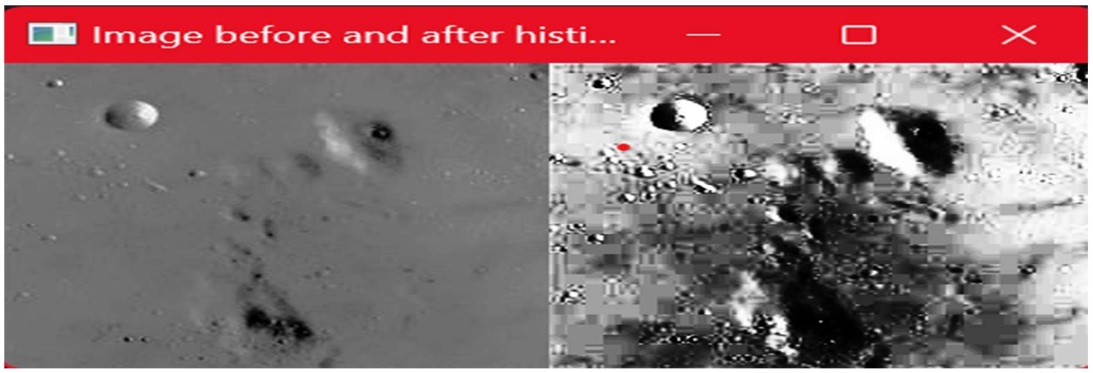
<u>Predefined:</u>

<u>Histograms</u>:



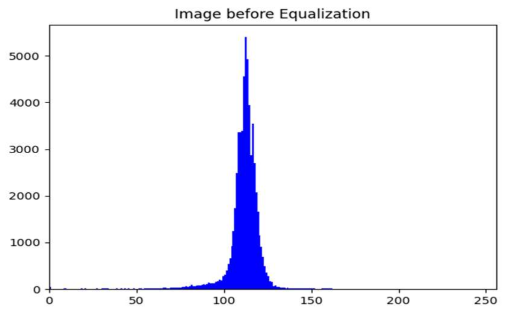


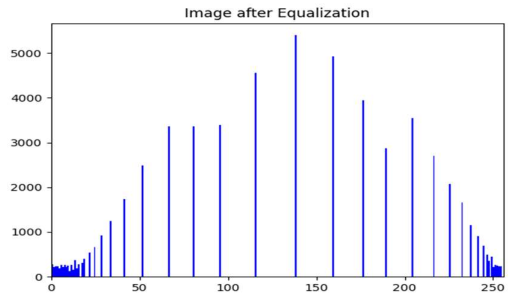
<u>Images</u>:



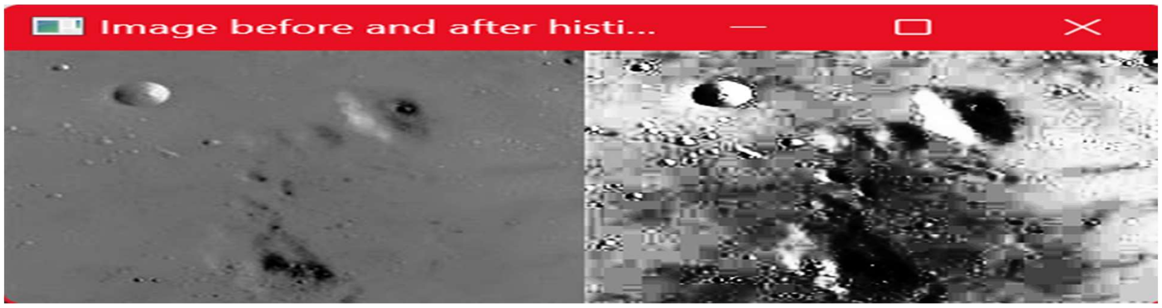
<u>User Defined</u>:

<u>Histogram</u>





<u>Images</u>:



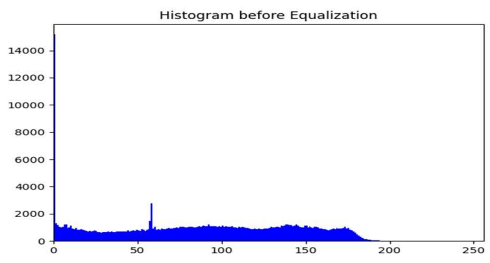
For Color:

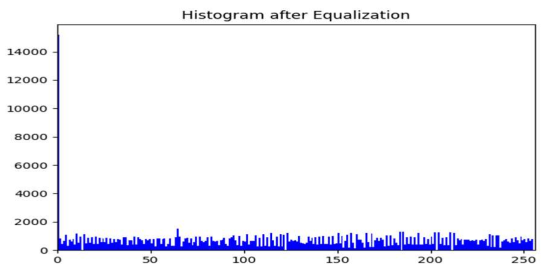
```python
import cv2
import numpy as np
from PIL import Image
from graph_helper import plot_graph

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:

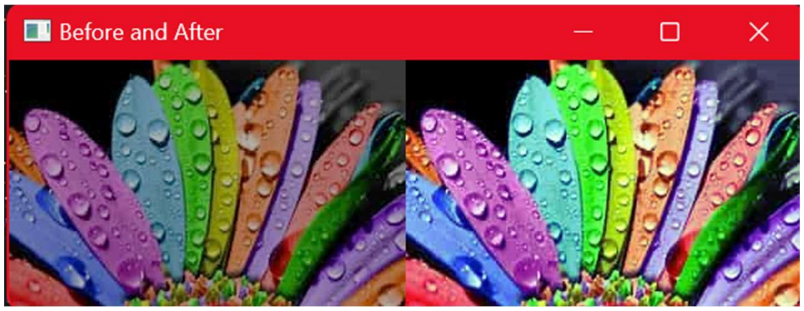
Predefined:

Histograms:





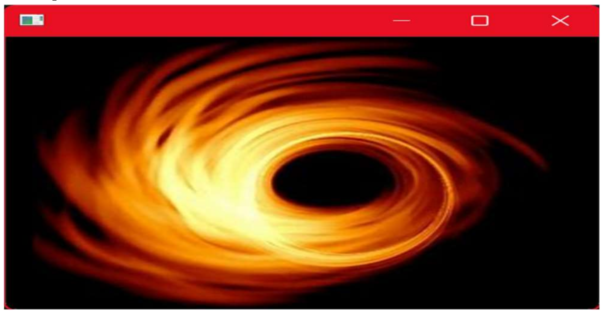
Images:

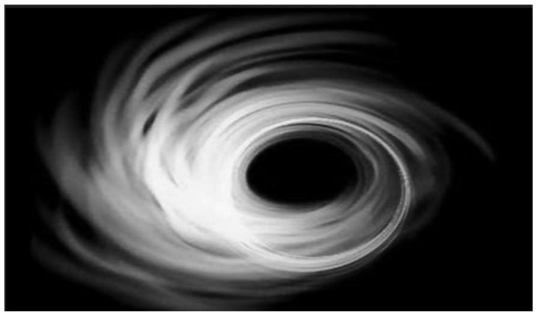


2. Convert rgb2gray to convert it from RGB to Monochrome
Code:

```
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:

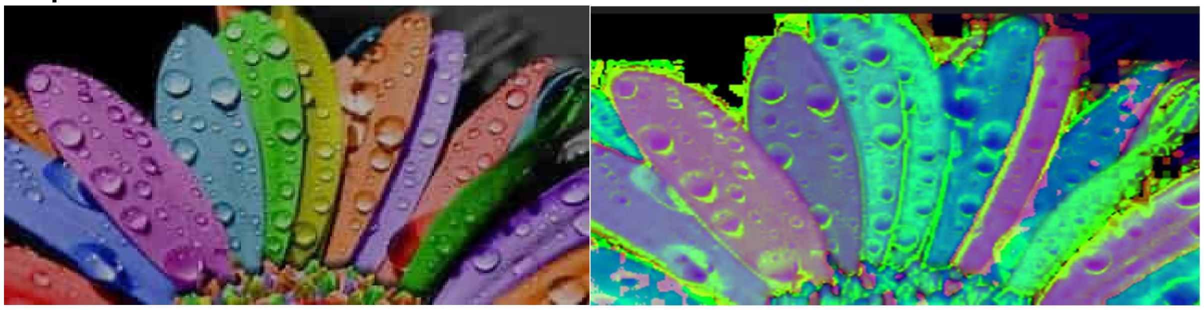




3. Make use of RGB2HSI and experiment it in HIS color model
   Code:

```python
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:

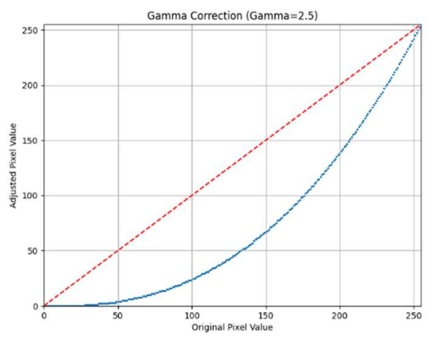


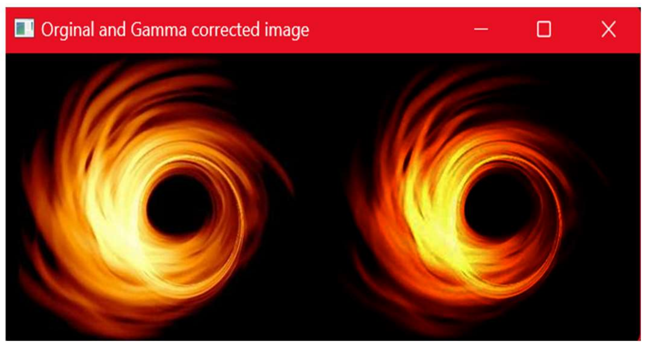
4. Make it poor contrast:
   a. Dark Image (Subtract/Gamma correction), Bright Image (Add/Gamma Correction)
      Code:

```python
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)
```

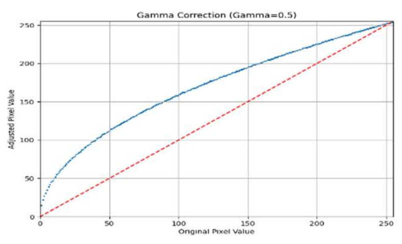
Output:

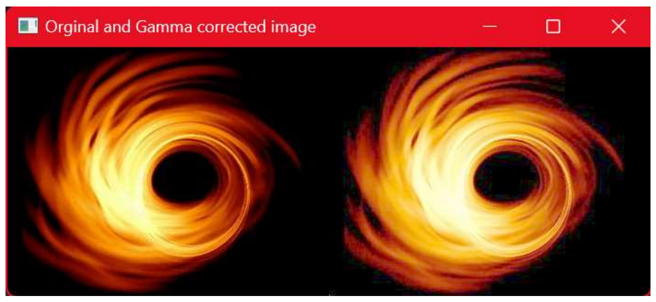
<u>Brighter Image: -(Gamma 2.5)</u>





<u>Lighter Image: - (Gamma 0.5)</u>

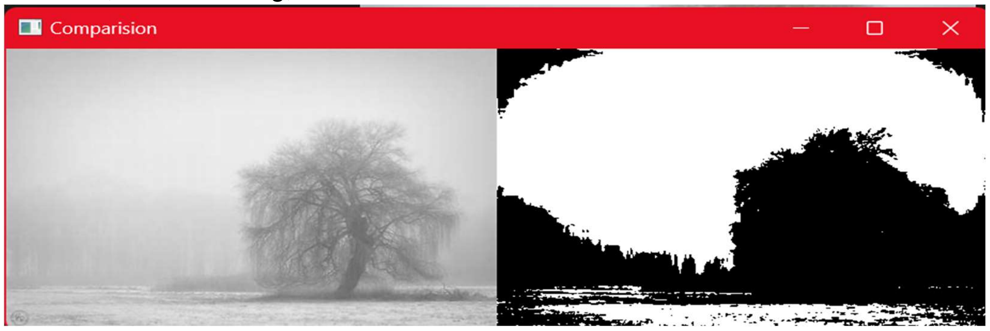




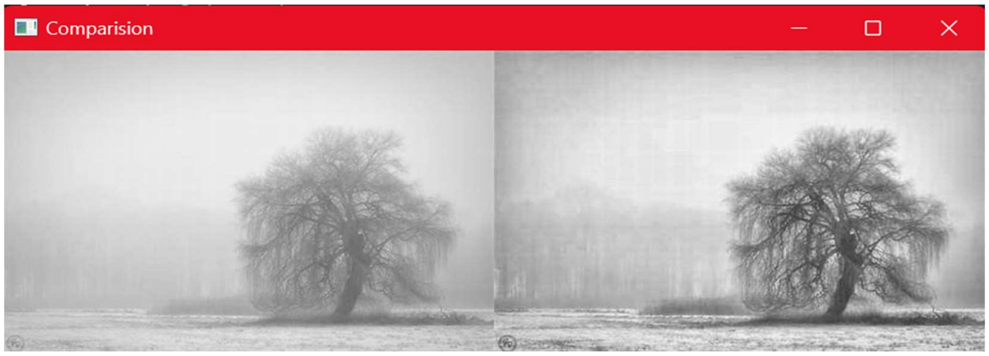
b. Low contrast (Thresholding) , High Contrast (a high contrast image usually exhibits a bimodal histogram with clear separation between the two predominant modes)
Code:

```
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)



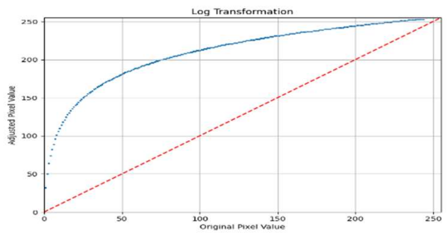
High Contrast: (CLAHE)

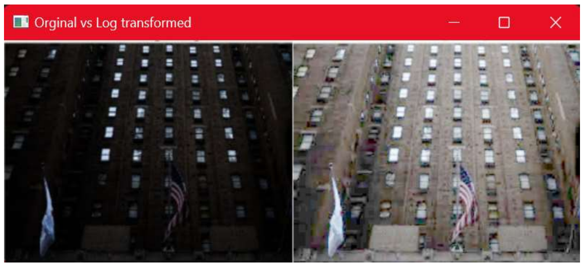


5. Write a program on Log transformation, Gamma correction, image negative and contrast stretching. You may Run your algorithm on MATLAB/PYTHON and print out the histogram on the screen, or plot it. A.
   a. **Log Transformation**
      Code:

```
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:





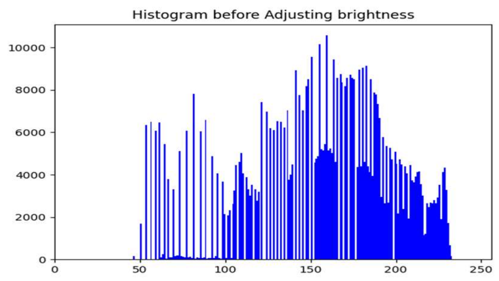
b. Gamma Correction

Code

Same as brightness adjustment code in 4a

Output

Histogram:



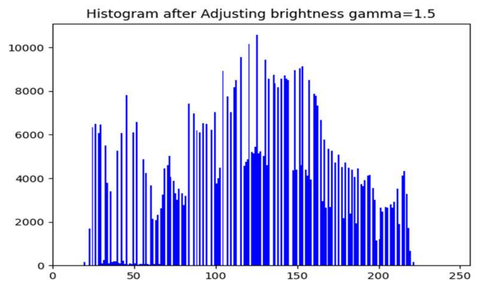
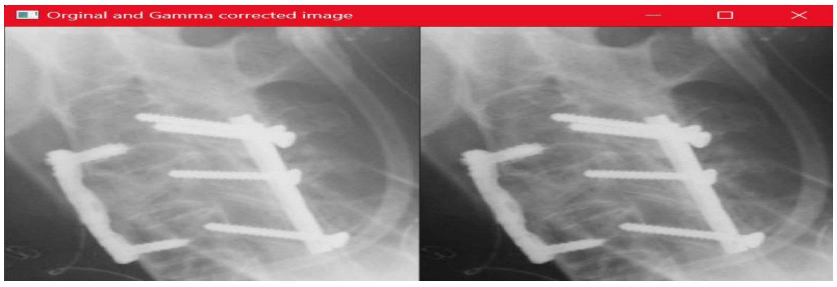


Image:

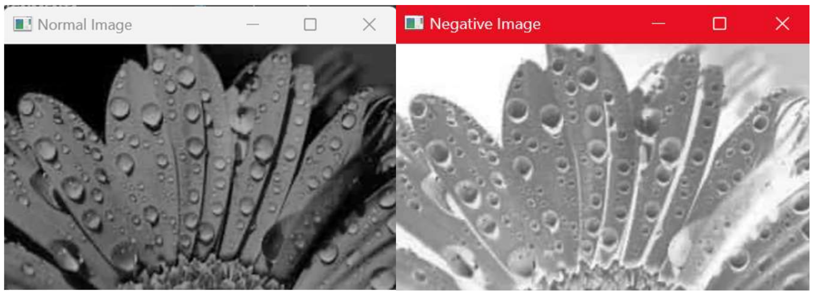


c. Image Negative

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:



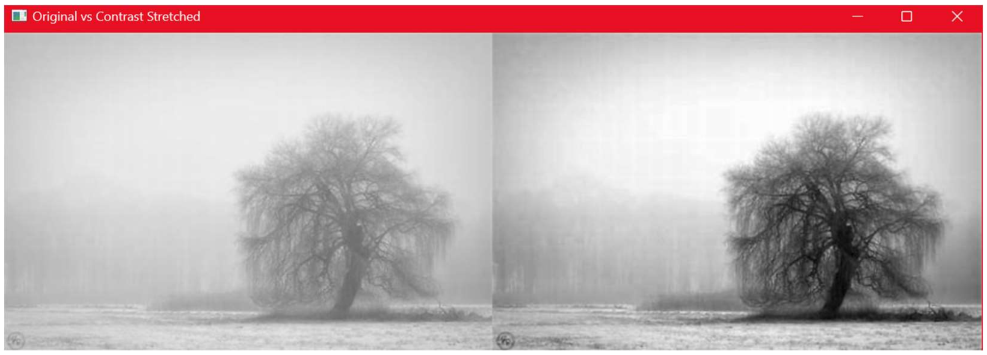
d. Contrast Stretching

Code:

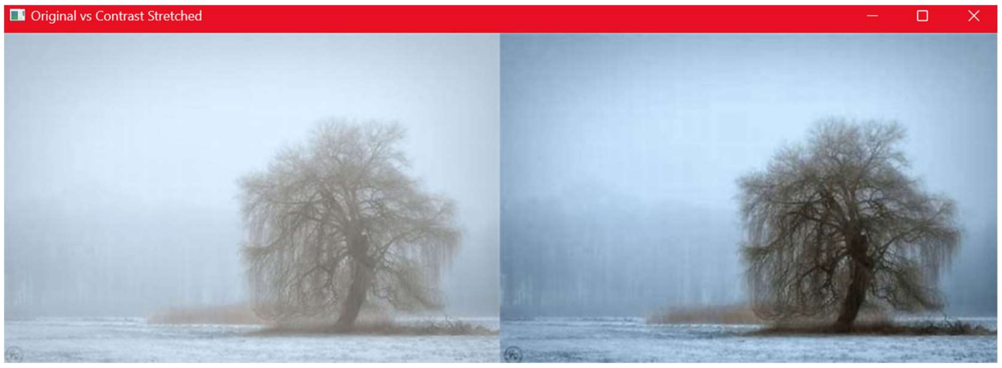
```
import cv2
import numpy as np

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:


Original vs Contrast Stretched