CEN 472
Digital Image Processing Lab
Journal - Lab 7

Name: **M MUNEEB AHMED KIANI**

Enrollment #: **01-135212-063**

Class: **BSCS 7B**

Objective

The purpose of today's lab is to have an insight into the gray intensity slicing, histogram equalization and contrast stretching.

Gray-Level Slicing

Highlighting a specific band (or range) of gray-intensities in an image is referred as gray-level slicing. The purpose of gray-level slicing is to assign more weight to certain details/information in an image for the purpose of analysis or to make them more visible in an image. Examples of typical gray level slicing can be seen in the following figure.

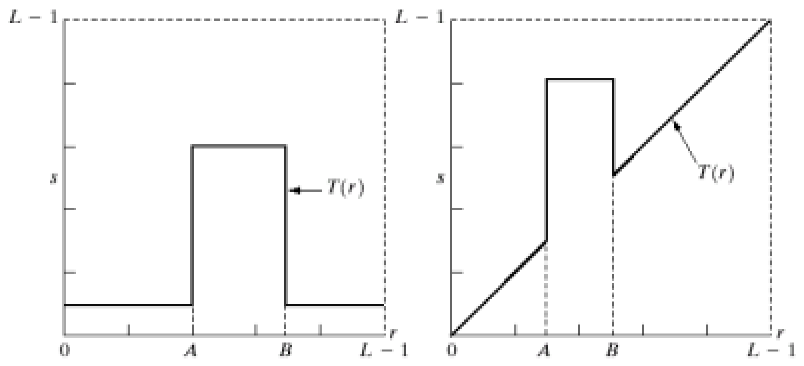


Figure 1: Gray level slicing transformations

The following two exercises will help you better understand the concept of gray level slicing.

Exercise 1

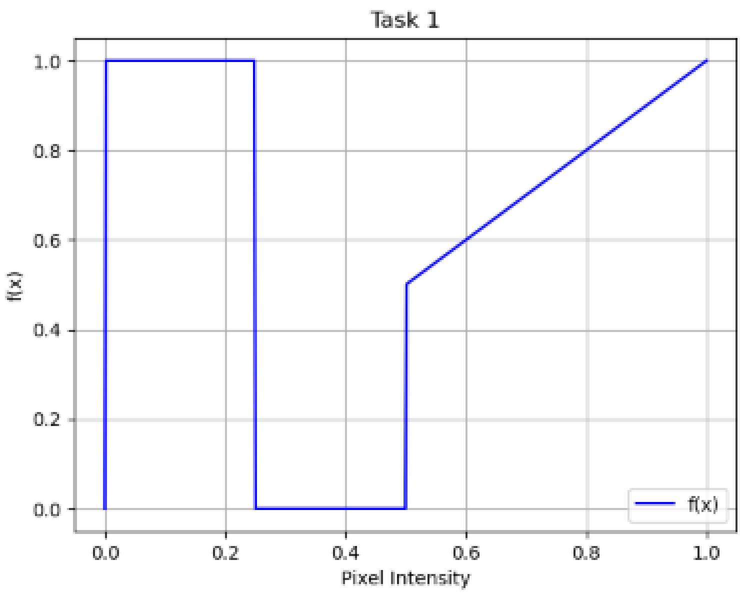
Consider a function f(x) defined over an image with the intensities in the range [0 1], as defined below:

$$f(x) = \begin{cases} 1 - x & 0 < x \leq 0.25 \\ 1 & 0.25 < x \leq 0.5 \\ x & x > 0.5 \end{cases}$$

a) Draw a graph for f(x) to show its influence on image intensities.

b) Write a Python program to implement f(x) on an input image. Also, show your result after transformation.

```python
import numpy as np
import matplotlib.pyplot as plt

# Define the transformation function f(x)
def f(x):
    if 0 < x <= 0.25:
        return 1
    elif 0.25 < x <= 0.5:
        return 0
    else:
        return x

x_values = np.linspace(0, 1, 500)

y_values = np.array([f(x) for x in x_values])


plt.plot(x_values, y_values, label="f(x)", color='blue')
plt.title("Task 1")
plt.xlabel("Pixel Intensity")
plt.ylabel("f(x)")
plt.grid(True)
plt.legend()
plt.show()
```



Histogram Equalization

Histogram equalization is the process of re-allocating intensity values of the pixels in an image such that the output image contains uniform distribution of intensities defined by a monotonically increasing function T(r).

Function in Python:

`cv2.equalizeHist(gray):` equalizes histogram of an input image in_img for the specified gray levels L.

How to Equalize a Histogram?

Consider an image of order 4x4 with 24 = 16 (0 – 15) gray levels. The histogram and the cumulative histogram of the image are also shown in the following.

k	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
n_k	0	1	3	1	1	1	1	2	1	3	1	0	1	0	0	0
cum. n_k	0	1	4	5	6	7	8	10	11	14	15	15	16	16	16	16

Original image:

2	5	9	9
7	6	2	2
8	9	7	3
4	10	12	1

4	7	14	14
10	8	4	4
11	14	10	5
6	15	16	1

To Normalize the intensity values use : round **[(L-1) * (s/ (M x N))]**

Where MXN is the size of the image.

Algorithm to Equalize an Image Histogram

1. Compute histogram, as follows:

   h(in_image(i, j) + 1) = h(in_image(i, j) + 1) + 1;

2. Compute cumulative histogram, as follows:

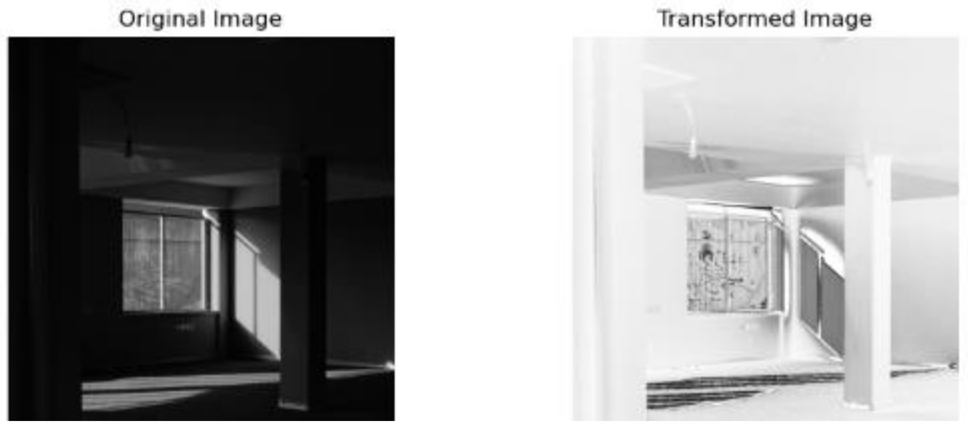
   cum_h(1) = h(1);

   cum_h(k) = cum_h(k - 1) + h(k); *% for all k ≥ 2*

3. Copy intensities from in_image into a new image out_image, as follows:

   *out_image(i, j) = cum_h(in_image(i, j) + 1)/ (r*c);*

```python
import numpy as np
import cv2
import matplotlib.pyplot as plt

def f(x):
    if x < 0.25:
        return 1 - x
    elif x < 0.5:
        return 1
    else:
        return x

def apply_transformation(image):
    image_normalized = image / 255.0
    transformed_image = np.vectorize(f)(image_normalized)
    return np.clip(transformed_image * 255, 0, 255).astype(np.uint8)

input_image = cv2.imread('pavel-moiseev-hFmxJMnvECc-unsplash.jpg', cv2.IMREAD_GRAYSCALE)

if input_image is None:
    print("Error: Image not found.")
else:
    output_image = apply_transformation(input_image)

    plt.figure(figsize=(10, 5))
    plt.subplot(1, 2, 1)
    plt.title('Original Image')
    plt.imshow(input_image, cmap='gray')
    plt.axis('off')

    plt.subplot(1, 2, 2)
    plt.title('Transformed Image')
    plt.imshow(output_image, cmap='gray')
    plt.axis('off')

    plt.show()
```



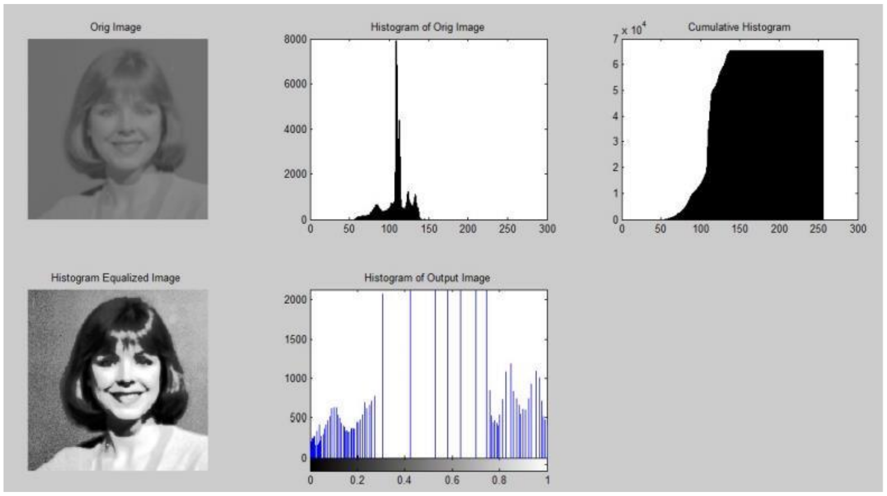
Exercise 2

Write a function named ‘myhist_equ’ to MANUALLY equalize histogram of an image and map corresponding results:

INPUTS: image

OUTPUTS: Display the input image as well as equalized (resultant) image along with histogram before and after histogram equalization.

Note: Do not use the `cv2.equalizeHist()` function in Python Load the image ‘person.png’ and display the output as illustrated in Figure 2.



```python
import cv2
import numpy as np
import matplotlib.pyplot as plt

def myhist_equ(image):

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


    cdf = hist.cumsum()
    cdf_normalized = cdf * hist.max() / cdf.max()
    cdf_m = np.ma.masked_equal(cdf, 0)
    cdf_m = (cdf_m - cdf_m.min()) * 255 / (cdf_m.max() - cdf_m.min())
    cdf = np.ma.filled(cdf_m, 0).astype('uint8')
    equalized_image = cdf[image]
    plt.figure(figsize=(10, 6))
    plt.subplot(2, 2, 1)
    plt.imshow(image, cmap='gray')
    plt.title('Original Image')
    plt.subplot(2, 2, 2)
    plt.hist(image.flatten(), 256, [0, 256], color='blue')
    plt.title('Original Histogram')
    plt.subplot(2, 2, 3)
    plt.imshow(equalized_image, cmap='gray')
    plt.title('Equalized Image')
    plt.subplot(2, 2, 4)
    plt.hist(equalized_image.flatten(), 256, [0, 256], color='red')
    plt.title('Equalized Histogram')
    plt.tight_layout()
    plt.show()
    return equalized_image
image = cv2.imread('pavel-moiseev-hFmxJMnvECc-unsplash.jpg', cv2.IMREAD_GRAYSCALE)
equalized_image = myhist_equ(image)
```

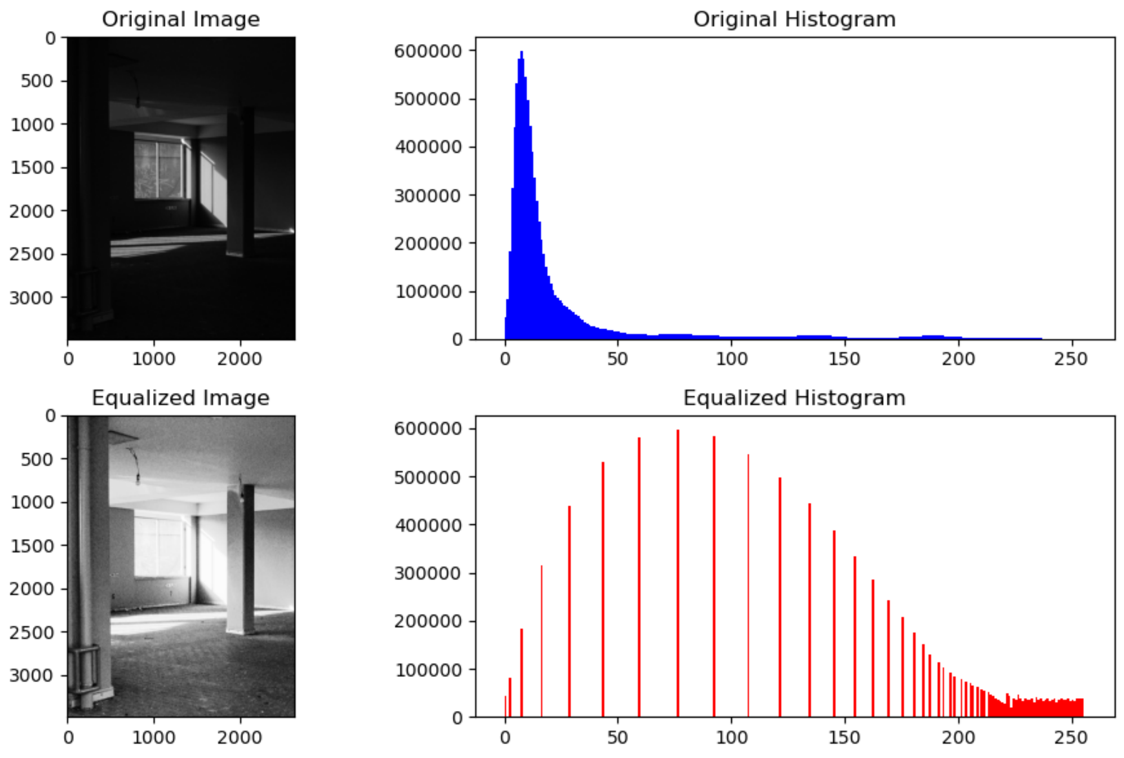


Figure 2:Histogram Equalization

Contrast Stretching

Contrast stretching attempts to improve an image by stretching the range of intensity values it contains to make full use of possible values. Unlike histogram equalization, contrast stretching is restricted to a linear mapping of input to output values. The result is less dramatic, but tends to avoid the sometimes artificial appearance of equalized images.

For each pixel, the original value r is mapped to output value s using the function:

$$s = T(r) = (r - r_{\min})\left[\frac{L-1}{r_{\max} - r_{\min}}\right]$$

Where rmin and rmax represent the minimum and the maximum gray values in the image, L-1 represents the maximum gray level in the image (for 8 bit images, L-1 = 255).

From the view point of implementation, convert the image to double, hence the factor L-1 = 1.

Exercise 3

Write a function named 'constrastStretch' to stretch the contrast of an image.

INPUTS: image

OUTPUTS: Display the input image as well as the contrast stretched (resultant) image along with the histogram of each. Load the image 'person.png' and display the output as illustrated in Figure 3.

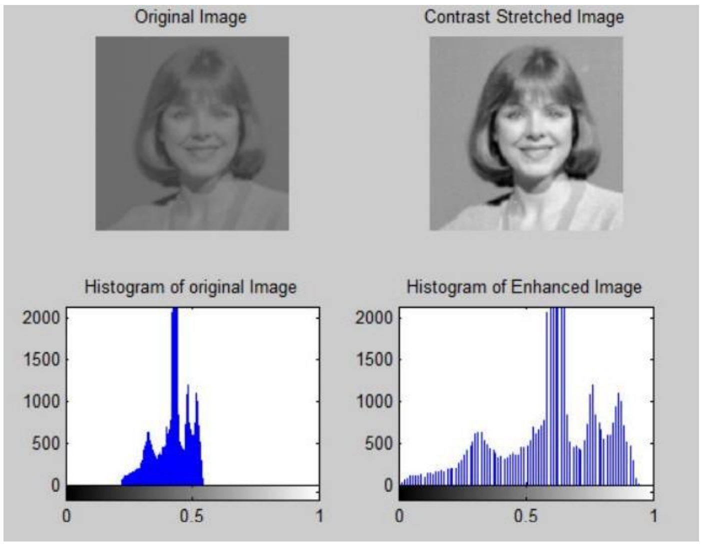
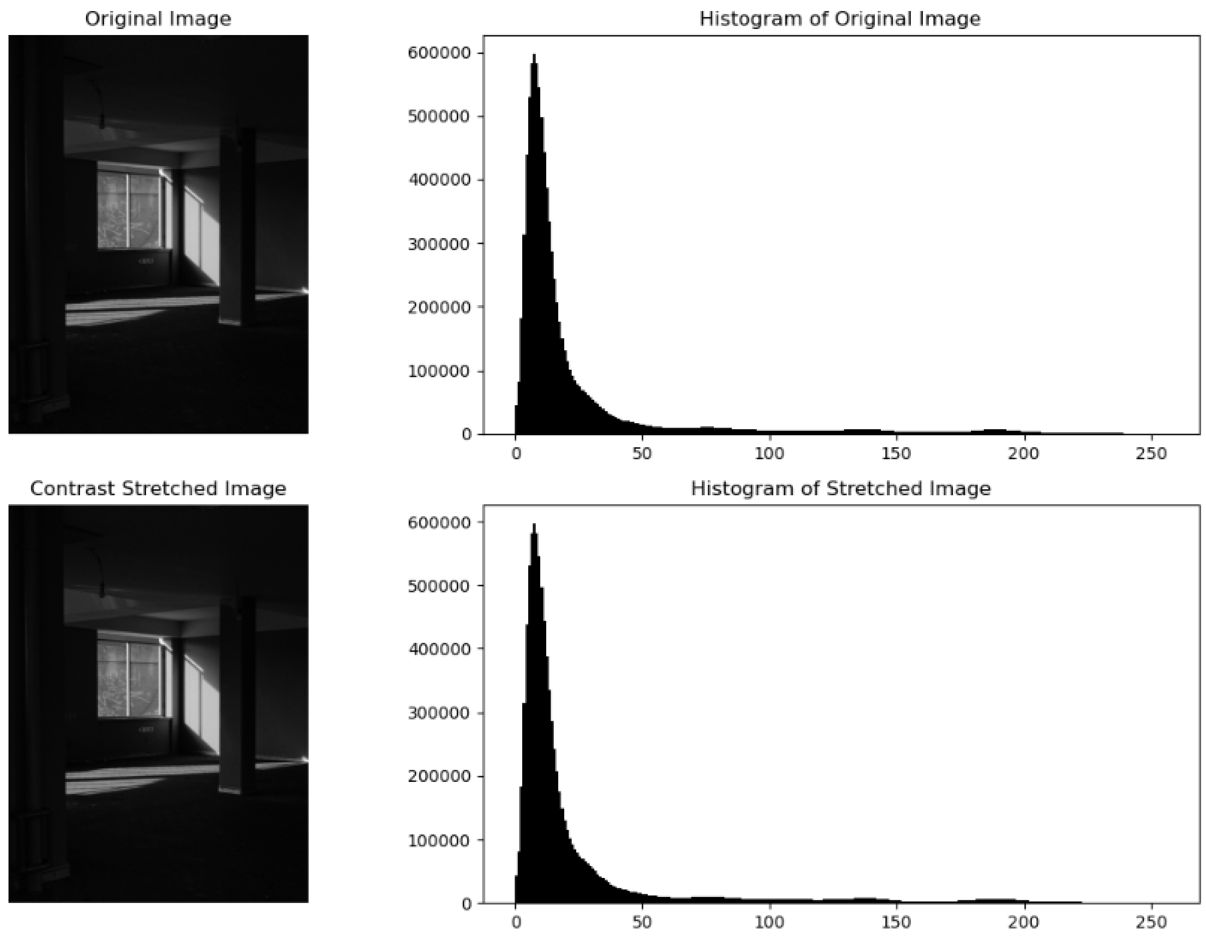


Figure 3:Contrast Stretching

```python
import numpy as np
import cv2
import matplotlib.pyplot as plt

def contrastStretch(image):
    min_pixel = np.min(image)
    max_pixel = np.max(image)
    stretched_image = (image - min_pixel) * (255 / (max_pixel - min_pixel))
    stretched_image = np.clip(stretched_image, 0, 255).astype(np.uint8)
    plt.figure(figsize=(12, 8))
    plt.subplot(2, 2, 1)
    plt.title('Original Image')
    plt.imshow(image, cmap='gray')
    plt.axis('off')
    plt.subplot(2, 2, 2)
    plt.title('Histogram of Original Image')
    plt.hist(image.flatten(), bins=256, range=[0, 256], color='black')
    plt.subplot(2, 2, 3)
    plt.title('Contrast Stretched Image')
    plt.imshow(stretched_image, cmap='gray')
    plt.axis('off')
    plt.subplot(2, 2, 4)
    plt.title('Histogram of Stretched Image')
    plt.hist(stretched_image.flatten(), bins=256, range=[0, 256], color='black')
    plt.tight_layout()
    plt.show()
input_image = cv2.imread('pavel-moiseev-hFmxJMnvECc-unsplash.jpg', cv2.IMREAD_GRAYSCALE)
if input_image is not None:
    contrastStretch(input_image)
else:
    print("Error: Image not found.")
```


Original Image
Histogram of Original Image
600000
500000
400000
300000
200000
100000
0
0
50
100
150
200
250
Contrast Stretched Image
Histogram of Stretched Image
600000
500000
400000
300000
200000
100000
0
0
50
100
150
200
250