Date: 13/02/2021

CS 6643 - Computer Vision, Spring 2021 **Prof.James Fishbaugh**

PROJECT I: HISTOGRAM EQUALIZATION

Pruthviraj R Patil, Computer Science, MS.
New York University, Tandon School of Engineering
Email: prp7650@nyu.edu

1. Introduction

In every subdomain of Artificial Intelligence, most of the data models' efficiencies are determined by multiple factors. One of the biggest factors is the quality of data using which they get trained on. In the case of Computer Vision, where we interpret the visual world, a quality pictures' dataset is quintessential no matter how good the architecture of the data model is. But, nowadays data is outsourced from zillions of devices worldwide, which makes it difficult to be cleaned after mining it. Hence, whenever the image data model is created, the preprocessing techniques are applied to the data to enhance the image quality.

Images are represented by the means of a matrix in which each cell is composed of its intensity value. This makes it the most important feature used for image classification methods. Pre-processing over this feature using the Histogram Equalization method improves contrast in images by spreading out their most frequent intensity values in their pixels. This method is used to increase the global contrast of the images, which helps in identifying objects at lower intensity regions.

2. Description and Illustration

2.1 Description of the Histogram Equalization process

The process of Histogram Equalization relies upon harnessing the cumulative probability function (CDF). Hence, here we tend to give the linear trend to CDF of the input image. Figure 2.11 shows an illustration of how the histogram of an image is made to follow a linear trend by applying the transformation function over its CDF.

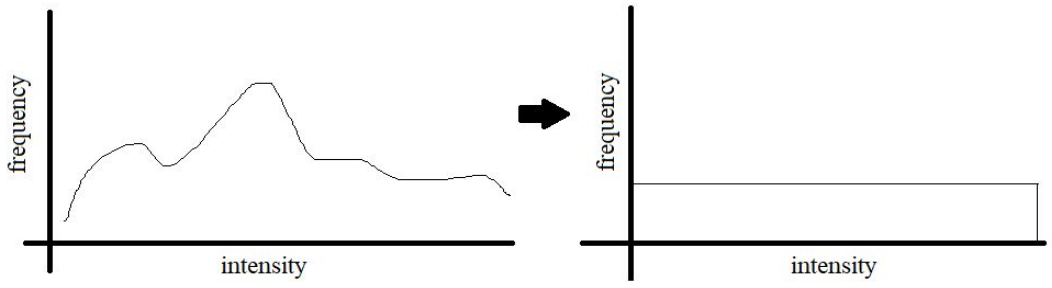

Fig 2.11Illustration how the histogram Equalization

CDF of the image histogram is a cumulative sum of all the probability of intensities of the image pixels which is nothing but the probability distribution function PDF. PDF is a function to get a probability of a pixel value x at a particular point. It is given by equation 2.11. and CDF are given by equations 2.11 and 2.12 respectively.

$$pdf(x) = \frac{Cardinality(x)}{\sum_{i=0}^{L-1} Cardinality(x_i)} = \frac{n_k}{\sum_{i=0}^{L-1} n_k(i)} \qquad \text{equation 2.11}$$

$$cdf(x) = \sum_{i=\infty}^{x} P(i) \qquad \text{equation 2.12}$$

Here, n_k is the frequency of the intensity x of a particular pixel at position i. $P(i)$ is the probability of the intensity at that particular pixel i derived using equation 2.11. Further, the new PDF is implemented over the intensities ranging from 0 to L-1 using equation 2.13. This makes the histogram of the old image follow the linear trend and spreads the intensities.

$$pdf^{|}(x) = (L-1)cdf(x) \qquad \text{equation 2.13}$$

2.2 Illustration of the process using an image

Let's consider a picture shown in the left half of figure 2.21. The initial image was of the dimensions (600, 800, 3). Where there are three channels of RGB. The image had to be converted into a grayscale image to make it 2 dimensional (600, 800). The inbuilt library of cv2 is used to import the image and convert it into grayscale.

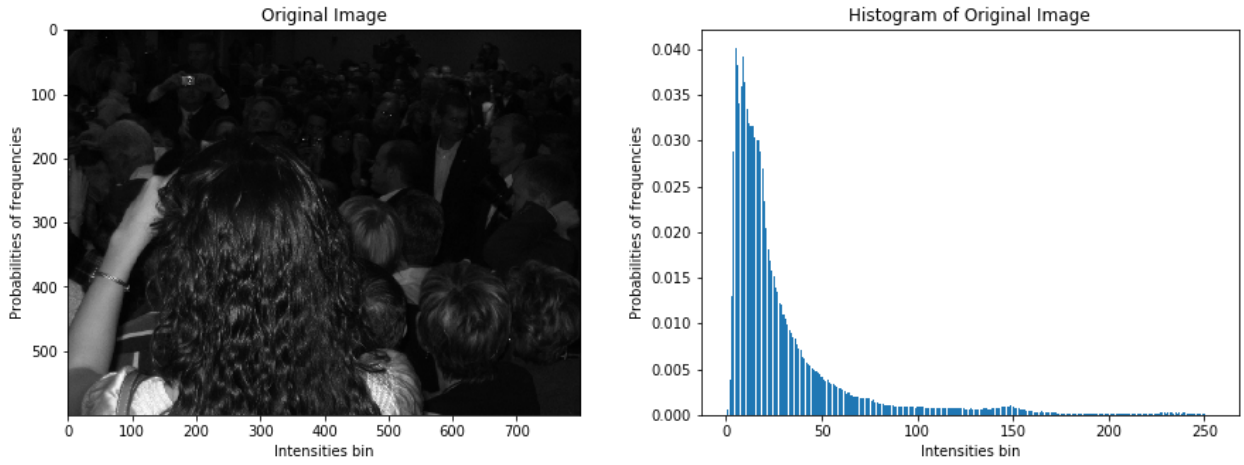

Figure 2.21 Crowd.png and its corresponding histograms (PDFs)

Now, the image is passed on to the Histogram Equalization pipeline. This pipeline can be seen in figure 2.22. Here, the intensity values are from the range of 0 to 255.

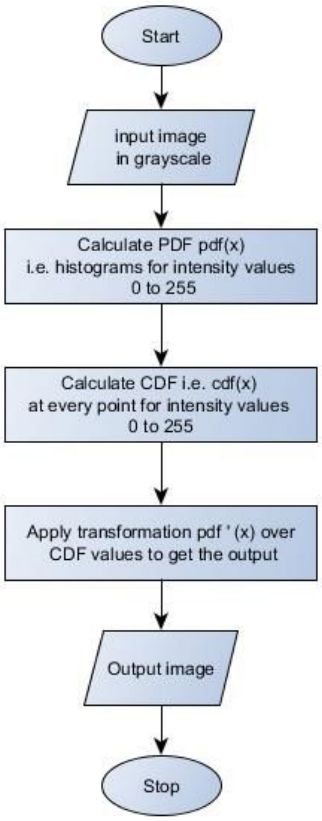

Fig: 2.22 Flowchart of the process

The probability distribution or the Histogram of the intensity values are derived using the probability distribution function over the intensity values, as shown in figure 2.21 (on the right side). Now, the CDF is applied over it and it is shown in 2.23.

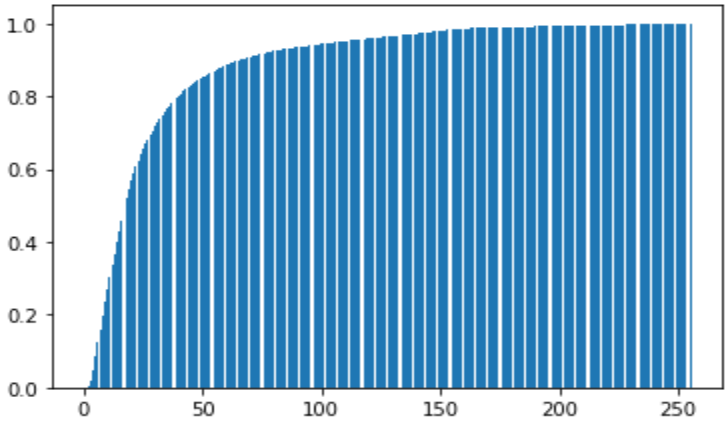

Figure 2.23 CDF of the input image

As we can see that the maximum value of the CDF is 1 as it is the cumulative sum of the pdf. Whereas the total sum of all the probabilities of histograms is as expected 1. Now, when the image is applied by the Histogram Equalization transformation, then the intensity values of the image gets spread out and hence refines the contrast of the image. The output of the Histogram transformation is shown in figure 2.24 on the left-hand side whereas the transformed probability distribution of the output image is shown on the right-hand side of the same figure.

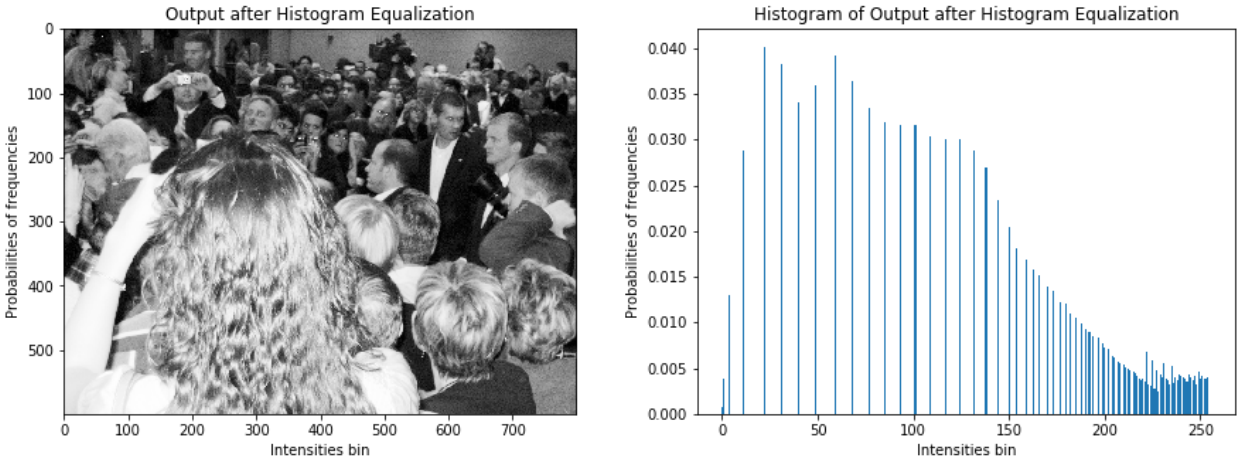

Figure 2.24 Transformed Crowd.png and its corresponding histograms (PDFs)

The change in the probability distribution is that the intensity frequencies are spread and are just as similar to the change shown in figure 2.11. As described earlier, the contrast of the image is refined and now, the background people in the image are visible clearly. This was possible because the process helped to contain the information of the positions of input image pixels in particular row and column but spread the intensities. It can be visualized easily in Figure 2.25 by it's CDF.

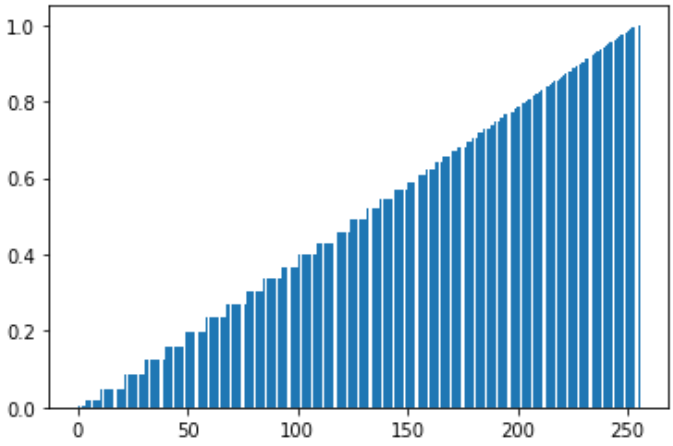

Figure 2.25 CDF of the output image

We can see that the linear trend is being followed by the output image. Even though it is not a completely accurate linear CDF looking at the initial part of the graph, it is due to the positions where there were higher slopes at the initial image CDF.

2.3 Re-Applying Histogram Equalization

As shown, the Histogram Equalization method refined the contrast of the image in a viable manner and this makes it organic to think that we could gain more refined contrast if we reapply the process over the output image.

But, as the linearity of the CDF is already maintained in the output image, the PDF of the output image also remains constant no matter how many times the Histogram Equalization process is applied over the output image. To verify this, the test run of applying the method over the output image from the initial equalization process is done. Figure 2.31 shows there is no change in the output image after the re-equalization process is carried on.

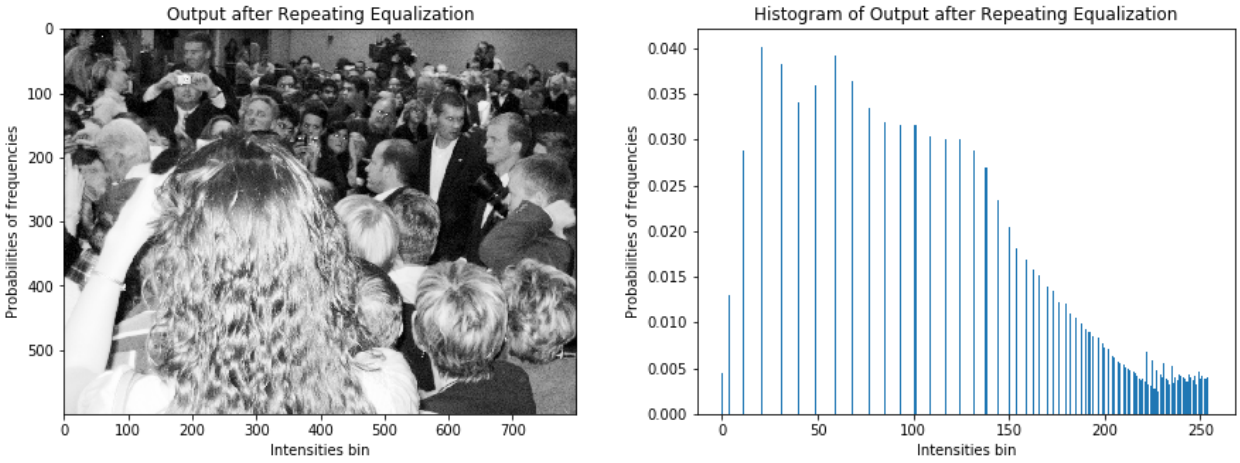

Figure 2.31 Transformed Crowd.png and its corresponding histograms (PDFs)

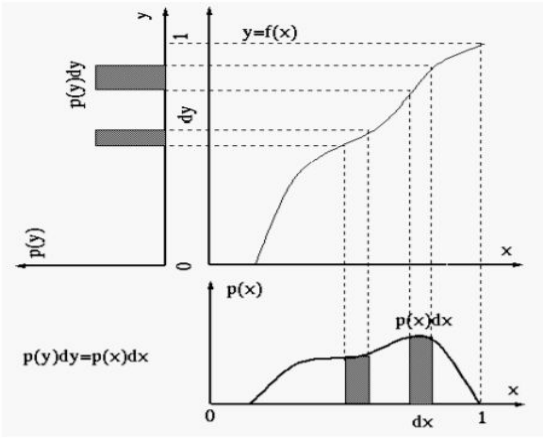

Figure 2.32 Pictorial representation of the process

If the PDF of the initial image be $p(x)$ whereas the transformation function after CDF is $y = f(x)$ whereas the resulting PDF after transformation is $p(y)$, then according to figure 2.31, equation $p(y)dy = p(x)dx$. But, in the case of the re-equalization process, the CDF of the transformed function states that $y = x$. Hence, the PDF of the final output image remains $p(x)$ itself.

2.4 Re-Applying Histogram Equalization

To test the effectiveness of the process, a low contrast image is taken into consideration. The image is given in Figure 2.41 and as we can see that it is a low contrast image. When the PDF of this image is taken, we can see that there is a higher amount of brightness over the image i.e. the frequency of pixels with higher intensities is higher than the ones with lower intensities. This makes the image look whiter than it is intended to be. This color correction can be done using Histogram Equalization process.

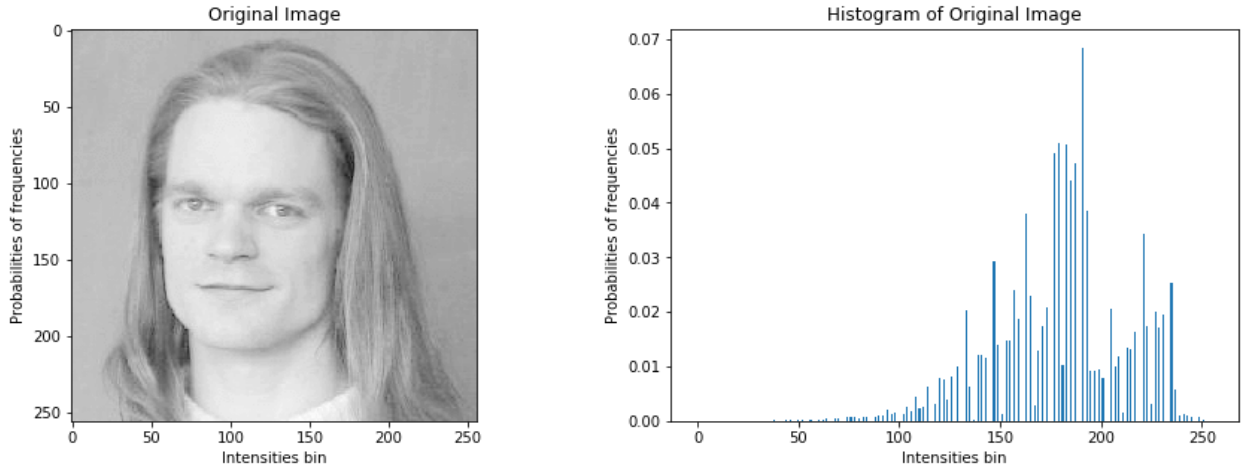

Figure 2.41 Low_contrast_img.png and its corresponding histograms (PDFs)

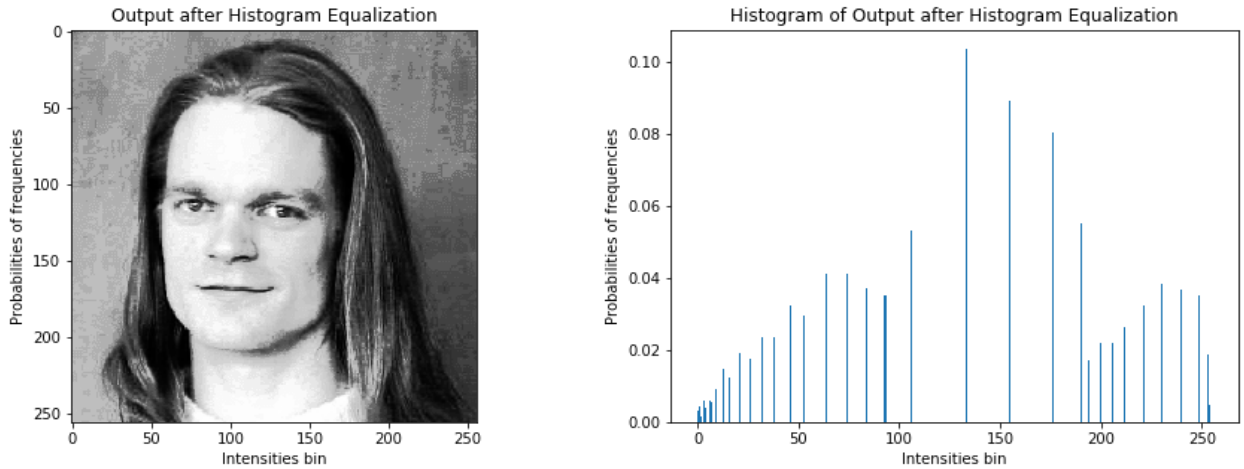

Figure 2.42 Transformed Crowd.png and its corresponding histograms (PDFs)

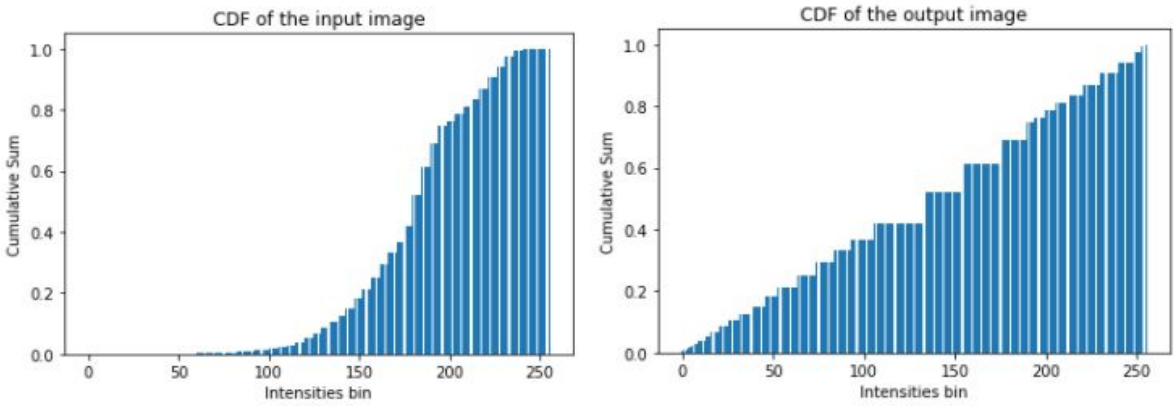

Figure 2.43 CDF of the input and output image respectively

The output, as expected, converted the low contrast image to the refined image as shown in Figure 2.42 and their respective CDF is shown in Figure 2.43. Here, the intensities of input image pixels are spread across approximately from 50 to 250. This makes the process easy to refine the contrast levels. But, if the input image had the intensities of it's pixels ranging in a very small interval, then the process wouldn't work. This is because the Histogram Equalization process focuses on Global Equalization of intensities. In such cases, we can go for methods where we localize this process by running it over the patches of the same image.

2.5. Applying Histogram Equalization over Patches of image.

The Histogram equalization process is mainly meant to globally transform all the pixels of the image. But, if we apply the same process to the patches of the image, this turns out to be a different result. The output depends on the input image's intensity distribution we can analyse this with 2 cases:

Case 1. Input image with evenly spreaded intensities:

These are the type of images where their PDFs show that the intensity bins populated are in the wide range. This shows that there are objects/information spread all over the image. In this case there would be a need for the global intensity refinement of the image. Hence, the local histogram equalization even though gives almost similar output as like the output of the global histogram equalization, this takes longer time as there would be a need to traverse over the overlapping image patches during iterations.

Case 2. Input image with intensity concentrated at one point:

These are the type of images where the background doesn't matter much but the foreground objects are of less intensity. In this case, there would be a need for the local histogram equalization process. Thus, if we divide the image into patches in this case (Say

50*50 pixels) and running the Histogram Equalization process over them in an overlapping manner, then the background intensities are affected less whereas the foreground object that needs to be intensified gains refinement in its local contrast as well as gains enhancements in the definitions of edges of the object in the image.

If the global Histogram Equalization method is applied over such objects, then even the information from the objects in the foreground too will get lost and the model that gets trained over such images tends to perform poorly. This could be checked by the local histogram equalization procedure.

III. Appendix - Python Code

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

#read image using path
def read_img(path):
    #status:complete
    img=cv2.imread(path)
    img_gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    img_gray = ((img_gray - np.min(img_gray)) * (1/(np.max(img_gray) -
np.min(img_gray)) * 255)).astype('uint8')
    return img_gray

# Create normalized intensity histogram from an input image
def create_pdf(im_in):
    #status:complete
    pdf=np.zeros(256)
    m=im_in.shape[0]
    n=im_in.shape[1]
    for i in range(m):
        for j in range(n):
            pdf[im_in[i][j]]+=1/(m*n)
    return pdf

# Create the cumulative distribution function from an input pdf
def create_cdf(pdf):
    #status:complete
    cdf=np.zeros(256)
    sums=0.0
    for i in range(256):
        sums+=pdf[i]
```

```
        cdf[i]=sums
    return cdf

# Create a histogram equalized image using your computed cdf
def histogram_equalization(im_in):
    #status:complete
    pdf = create_pdf(im_in) # previously implemented function
    cdf = create_cdf(pdf) # previously implemented function
    m=im_in.shape[0]
    n=im_in.shape[1]
    equalized_im=np.zeros((m, n))
    for i in range(m):
        for j in range(n):
            equalized_im[i][j]=cdf[im_in[i][j]]
    return equalized_im

#convert image to image with unsigned int clipped to 255 intensity
def convert_img_uint(img):
    #status:complete
    img = ((img - np.min(img)) * (1/(np.max(img) - np.min(img)) *
255)).astype('uint8')
    return img

#to juxtapose original image and final equalized image
def show_img_and_hist(img, name):
    #status:complete
    fig, ax= plt.subplots(nrows=1, ncols=2, figsize=(15, 5))
    ax[0].imshow(img, cmap="gray")
    ax[0].set_title(name)
    ax[0].set_xlabel("Intensities bin")
    ax[0].set_ylabel("Probabilities of frequencies")

    ax[1].bar(np.arange(256), create_pdf(img))
    ax[1].set_title("Histogram of "+name)
    ax[1].set_xlabel("Intensities bin")
    ax[1].set_ylabel("Probabilities of frequencies")

def main():
    #status:complete
    path='C:\\Users\\pruth\\Desktop\\NYU\\Year-1\\Spring\\CV\\crowd.png'
    img=read_img(path)
    equalized_im=histogram_equalization(img)
    equalized_im = convert_img_uint(equalized_im)
```

```
    show_img_and_hist(img, "Original Image")
    show_img_and_hist(equalized_im, "Output after Histogram Equalization")

    #reapplying the Histogram equalization for the corrected image.
    re_equalized_im=histogram_equalization(equalized_im)
    show_img_and_hist(equalized_im, "Equalized Image")
    re_equalized_im = convert_img_uint(re_equalized_im)
    show_img_and_hist(re_equalized_im, "Output after Repeating
Equalization")



    #applying the Histogram equalization over a low contrast image.

path_new_img='C:\\Users\\pruth\\Desktop\\NYU\\Year-1\\Spring\\CV\\low_contr
ast_img.png'
    lc_img=read_img(path_new_img)
    equalized_im_lc=histogram_equalization(lc_img)
    equalized_im_lc= convert_img_uint(equalized_im_lc)
    show_img_and_hist(lc_img, "Original Image")
    show_img_and_hist(equalized_im_lc, "Output after Histogram
Equalization")

if __name__ == "__main__":
    main()
```