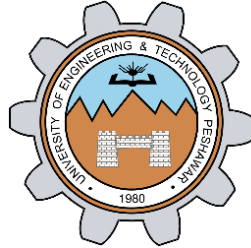


# Assignment 02



**Spring 2024**

## **Digital Image Processing**

Submitted by:

**Safi Ullah Khan (20pwcse1943)**

**Class Section: B**

“On my honor, as student of University of Engineering and Technology, I have neither given nor received unauthorized assistance on this academic work.”

Submitted to:

Dr. Abeer Irfan

(May 24, 2024)

**Department: Computer System Engineering**

|                                  |    |
|----------------------------------|----|
| Figure 1 code.....               | 4  |
| Figure 2 code2.....              | 5  |
| Figure 3 code3.....              | 5  |
| Figure 4 output1.....            | 6  |
| Figure 5 output2.....            | 6  |
| Figure 6 code4.....              | 7  |
| Figure 7 code5.....              | 7  |
| Figure 8 outpu4.....             | 8  |
| Figure 9 output5.....            | 8  |
| Figure 10 output6 .....          | 9  |
| Figure 11 output7 .....          | 9  |
| Figure 12 histocode.....         | 10 |
| Figure 13 histogram output ..... | 10 |

### *What are Histograms?*

A histogram is an approximate representation of the distribution of some numerical data. It is the most commonly used graph to show frequency distributions. It looks very much like a bar chart.

In terms of image:

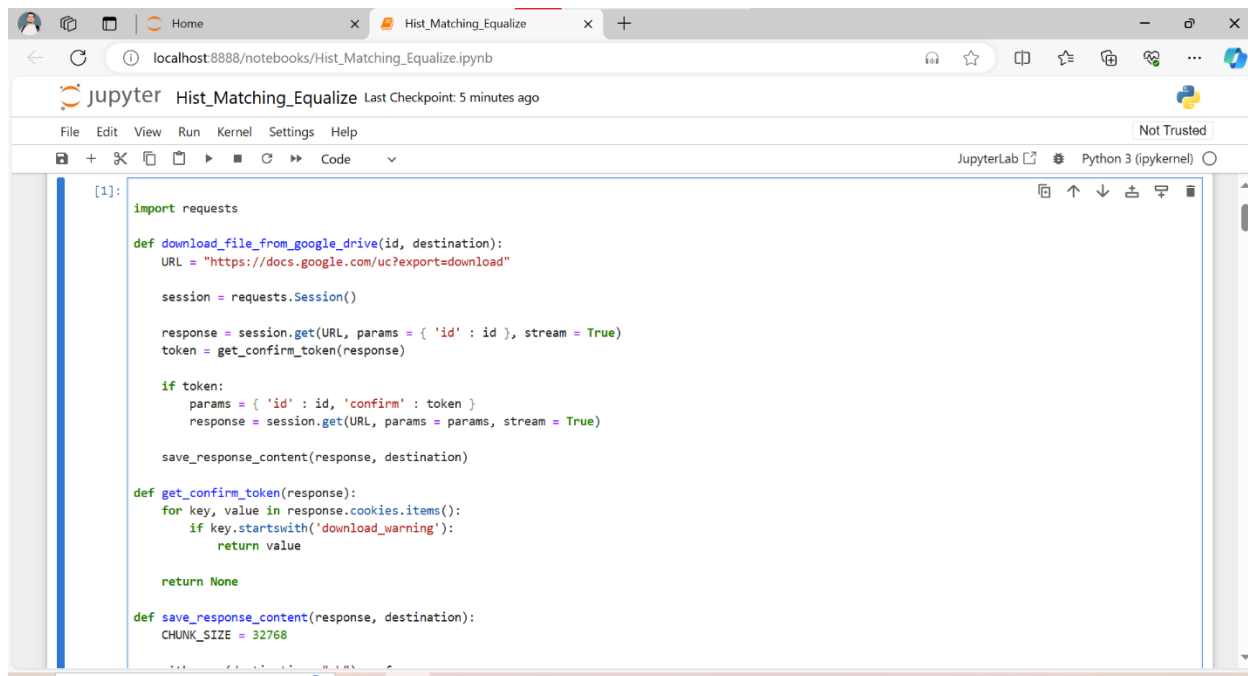
Histogram is a graphical representation of the intensity distribution of an image. In simple terms, it represents the number of pixels for each intensity value considered.

### *Histograms applications*

1. Histograms are preferred in applications, when there is a need for any of the following:
2. The data are numerical and the distribution of data is needed to be observed, especially when determining whether the output of a process is distributed approximately normally or not
3. Seeing whether a process change has occurred from one time period to another
4. Determining whether the outputs of two or more processes are different
5. Statistical properties need to be modeled
6. Some specific use cases of histograms are
7. In hydrology the histogram and estimated density function of rainfall are used to gain insight in their behaviour and frequency of occurrence.
8. In many Digital Image processing programs there is a histogram tool, which shows you the distribution of the contrast / brightness of the pixels.

### *Histogram Equalizing*

Histogram equalization is a method for contrast adjustment using the image's histogram. Basically, it is a computer image processing technique used to improve contrast in images. It accomplishes this by effectively spreading out the most frequent intensity values. When the useable data is represented by near contrast values, this approach generally boosts the global contrast of pictures. This enables locations with poor local contrast to gain a higher contrast.



The image shows a JupyterLab web interface in a browser. The address bar shows 'localhost:8888/notebooks/Hist\_Matching\_Equalize.ipynb'. The JupyterLab header includes the logo, the notebook name 'Hist\_Matching\_Equalize', and a 'Last Checkpoint: 5 minutes ago' status. Below the header is a menu bar with 'File', 'Edit', 'View', 'Run', 'Kernel', 'Settings', and 'Help'. To the right of the menu bar is a 'Not Trusted' warning. Below the menu bar is a toolbar with icons for file operations and a 'Code' dropdown menu. The main area contains a code editor with the following Python code:

```
[1]:
import requests

def download_file_from_google_drive(id, destination):
    URL = "https://docs.google.com/uc?export=download"

    session = requests.Session()

    response = session.get(URL, params = { 'id' : id }, stream = True)
    token = get_confirm_token(response)

    if token:
        params = { 'id' : id, 'confirm' : token }
        response = session.get(URL, params = params, stream = True)

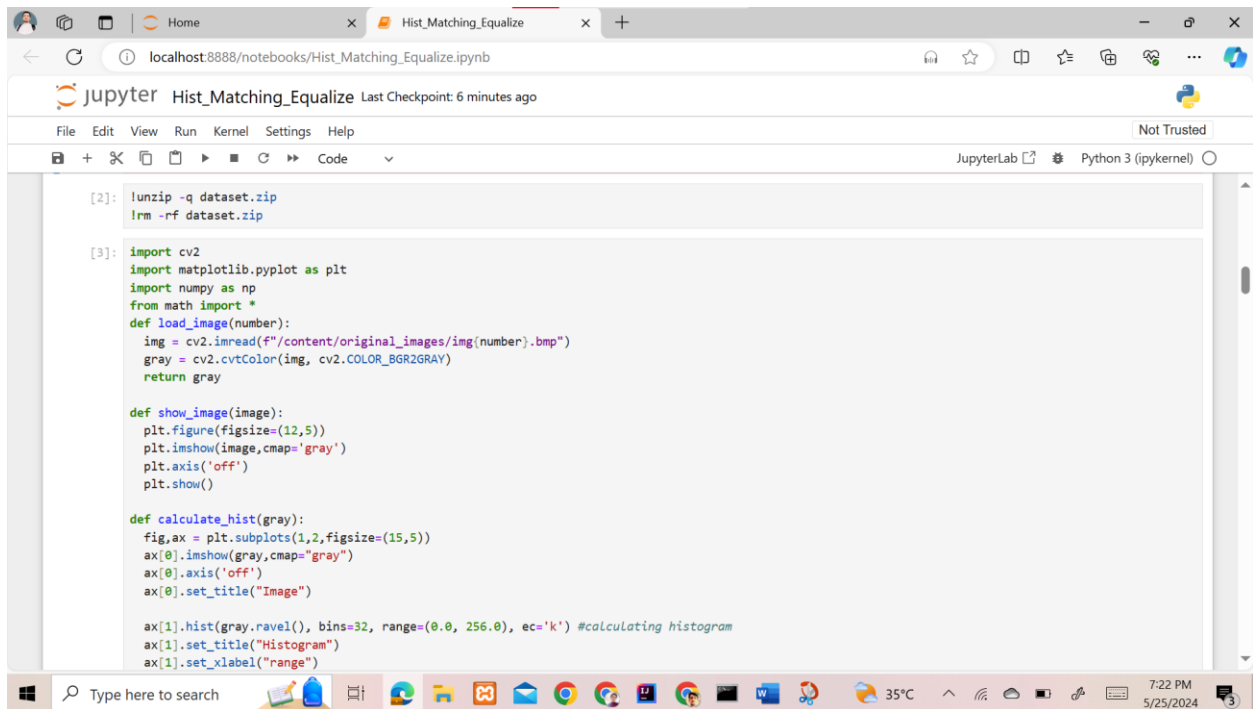
    save_response_content(response, destination)

def get_confirm_token(response):
    for key, value in response.cookies.items():
        if key.startswith('download_warning'):
            return value

    return None

def save_response_content(response, destination):
    CHUNK_SIZE = 32768
```

Figure 1 code



The screenshot shows a JupyterLab notebook interface with a browser window at the top displaying 'localhost:8888/notebooks/Hist\_Matching\_Equalize.ipynb'. The notebook has a menu bar (File, Edit, View, Run, Kernel, Settings, Help) and a toolbar. The code is as follows:

```
[2]: !unzip -q dataset.zip
    !rm -rf dataset.zip

[3]: import cv2
    import matplotlib.pyplot as plt
    import numpy as np
    from math import *
    def load_image(number):
        img = cv2.imread(f"/content/original_images/img{number}.bmp")
        gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
        return gray

    def show_image(image):
        plt.figure(figsize=(12,5))
        plt.imshow(image, cmap='gray')
        plt.axis('off')
        plt.show()

    def calculate_hist(gray):
        fig, ax = plt.subplots(1,2,figsize=(15,5))
        ax[0].imshow(gray, cmap="gray")
        ax[0].axis('off')
        ax[0].set_title("Image")

        ax[1].hist(gray.ravel(), bins=32, range=(0.0, 256.0), ec='k') #calculating histogram
        ax[1].set_title("Histogram")
        ax[1].set_xlabel("range")
```

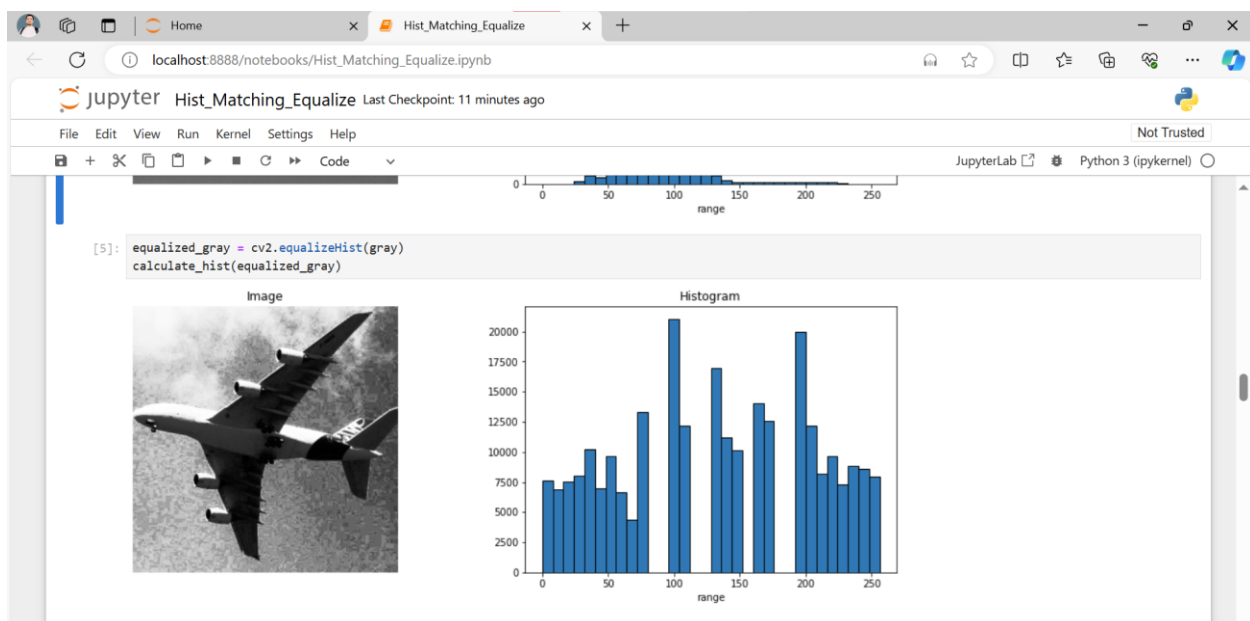
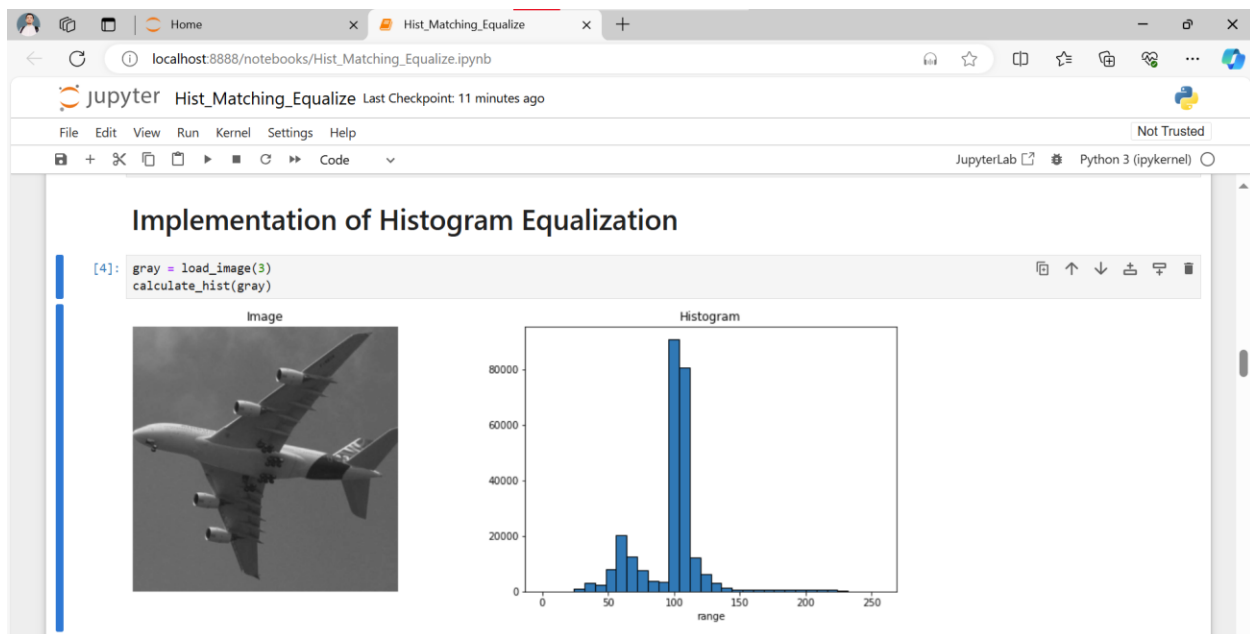
Figure 2 code2

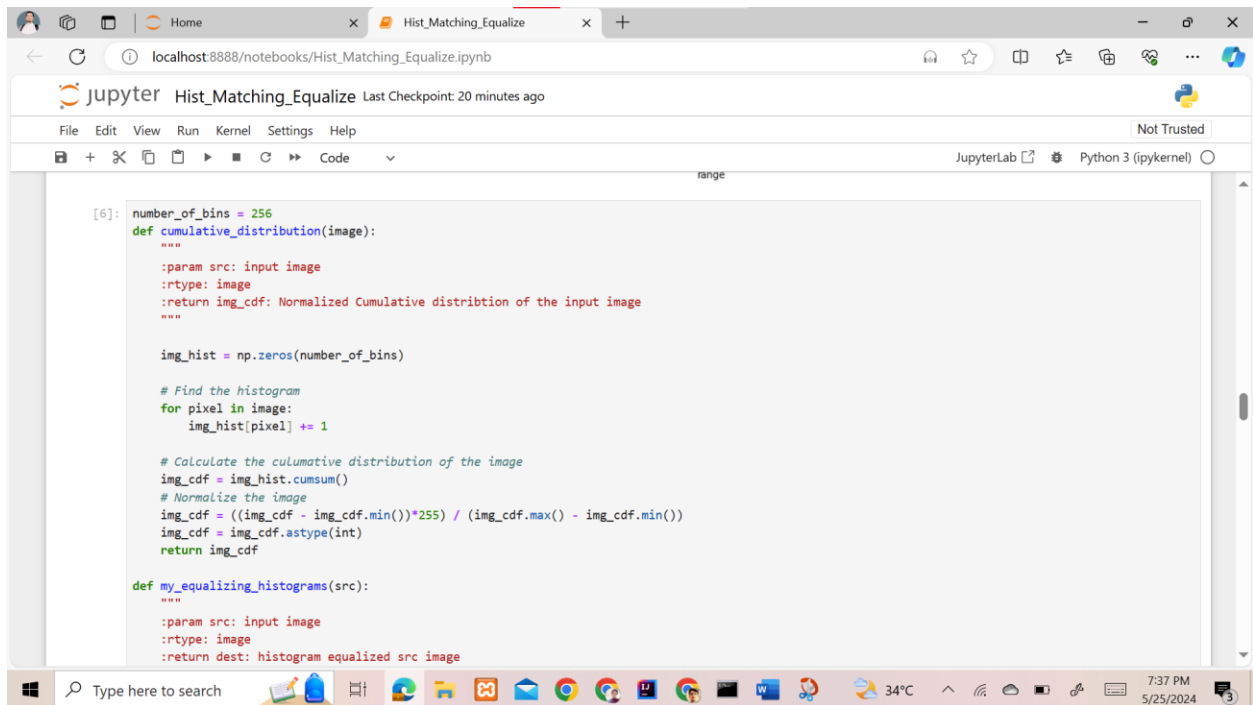
```
def compare_matched_hist(src,dst,matched_src):
    images = [src,dst,matched_src]
    headings = ["Source","Destination","Matched Source"]
    n,m = len(images),2
    fig,ax = plt.subplots(n,m,figsize=(15,10))

    for i, (heading,img) in enumerate(zip(headings,images)):
        ax[i,0].imshow(img,cmap="gray")
        ax[i,0].axis('off')
        ax[i,0].set_title(heading)

        ax[i,1].hist(img.ravel(), bins=32, range=(0.0, 256.0), ec='k') #calculating histogram
        plt.show()
```

Figure 3 code3





The image shows a JupyterLab window titled 'Hist\_Matching\_Equalize' with a 'Not Trusted' warning. The code in the cell is as follows:

```
[6]: number_of_bins = 256
def cumulative_distribution(image):
    """
    :param src: input image
    :rtype: image
    :return img_cdf: Normalized Cumulative distribution of the input image
    """

    img_hist = np.zeros(number_of_bins)

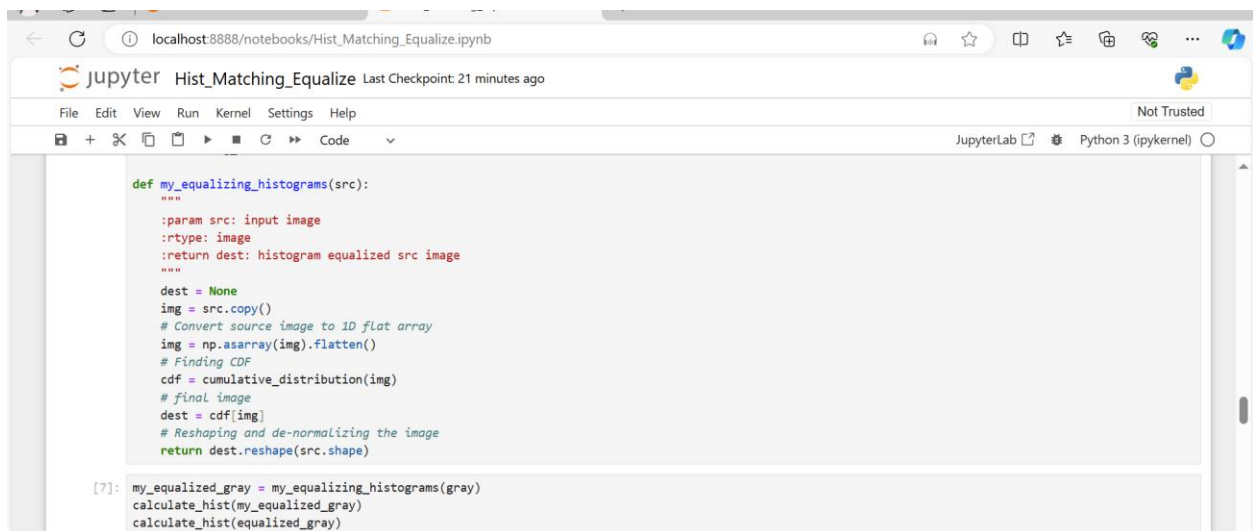
    # Find the histogram
    for pixel in image:
        img_hist[pixel] += 1

    # Calculate the cumulative distribution of the image
    img_cdf = img_hist.cumsum()
    # Normalize the image
    img_cdf = ((img_cdf - img_cdf.min())*255) / (img_cdf.max() - img_cdf.min())
    img_cdf = img_cdf.astype(int)
    return img_cdf

def my_equalizing_histograms(src):
    """
    :param src: input image
    :rtype: image
    :return dest: histogram equalized src image
    """
```

The Windows taskbar at the bottom shows the date as 5/25/2024 and the time as 7:37 PM.

Figure 6 code4



The image shows the same JupyterLab window with the following code:

```
def my_equalizing_histograms(src):
    """
    :param src: input image
    :rtype: image
    :return dest: histogram equalized src image
    """

    dest = None
    img = src.copy()
    # Convert source image to 1D flat array
    img = np.asarray(img).flatten()
    # Finding CDF
    cdf = cumulative_distribution(img)
    # final image
    dest = cdf[img]
    # Reshaping and de-normalizing the image
    return dest.reshape(src.shape)

[7]: my_equalized_gray = my_equalizing_histograms(gray)
calculate_hist(my_equalized_gray)
calculate_hist(equalized_gray)
```

Figure 7 code5

```

return dest.reshape(src.shape)

[7]: my_equalized_gray = my_equalizing_histograms(gray)
      calculate_hist(my_equalized_gray)
      calculate_hist(equalized_gray)

```

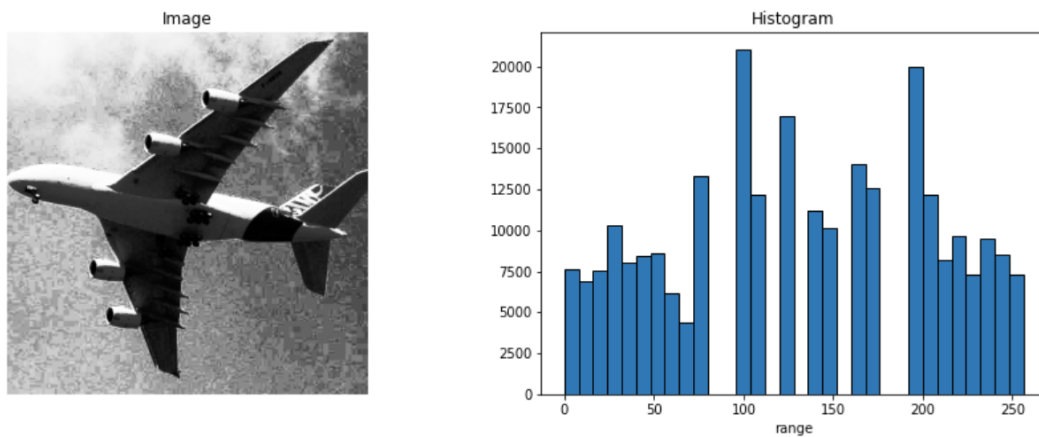


Figure 8 output4

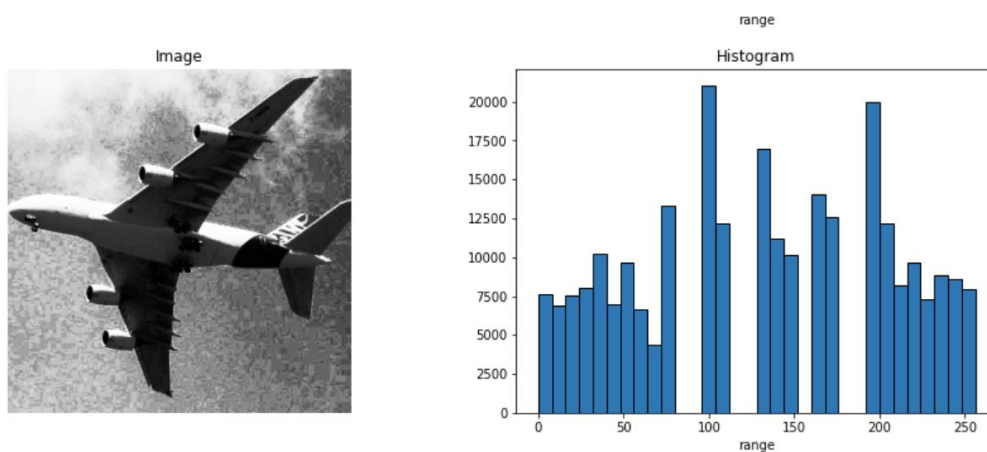


Figure 9 output5

## Histogram matching

Comparing with the inbuilt results, my custom implementation seems to be reproducing almost similar results. The image histogram of the inbuilt function and the custom implementation are almost similar. Apart from the histogram, the contrast of the image itself has been improved, and the implementation seems to be performing histogram equalization correctly.

Histogram matching is the transformation of an image so that its histogram matches a specified histogram. In order to match the histogram of images A and B, we need to first equalize the histogram of both images. Then, we need to map each pixel of A to B using the equalized histograms. Then we modify each pixel of A based on that of B.



Histogram matching may be used to balance detector responses. It can be used to equalise two pictures that were taken in the same place with the same local lighting (such as shadows), but with different sensors, atmospheric conditions, or global illumination.

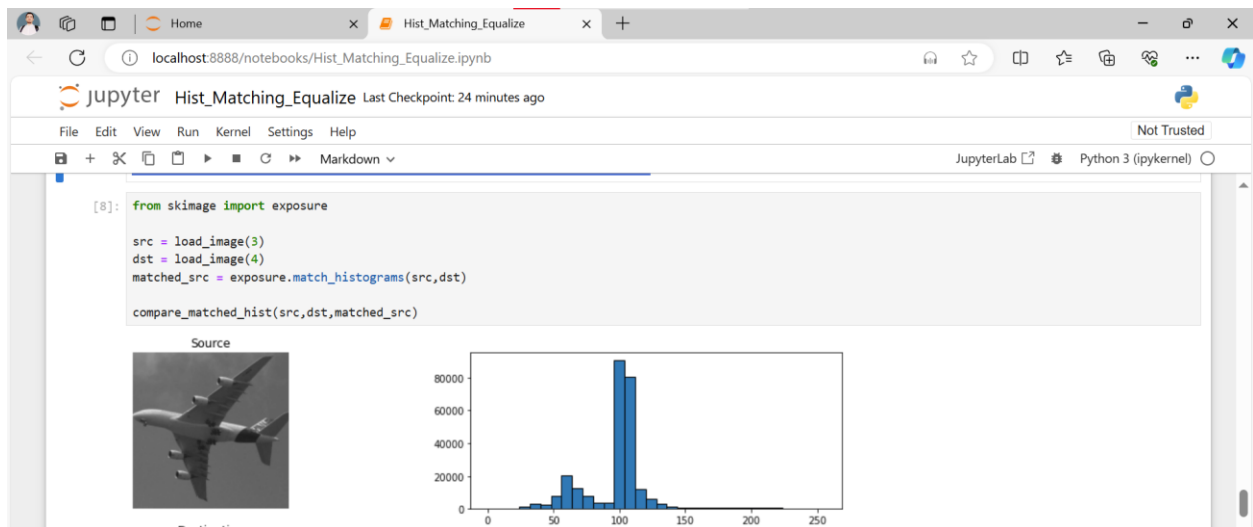


Figure 10 output6

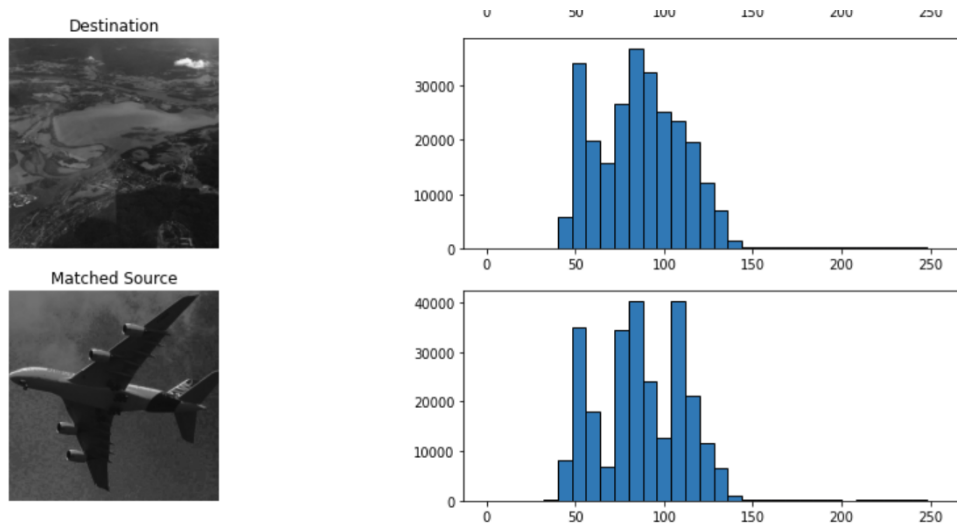


Figure 11 output7

```
[9]: def my_matching_histograms(src,dst):
    """
    :param src: input image
    :param dst: reference image
    :rtype: image
    :return matched_src: histogram matched src image
    """
    src_values, src_unique_indices, src_counts = np.unique(src.ravel(), return_counts=True, return_inverse=True)
    dst_values, dst_counts = np.unique(dst.ravel(), return_counts=True)

    # Normalizing cdf for source and destination images
    src_cdf = np.cumsum(src_counts) / len(src)
    dst_cdf = np.cumsum(dst_counts) / len(dst)

    matched_src = np.interp(src_cdf, dst_cdf, dst_values)
    return matched_src[src_unique_indices].reshape(src.shape)
```

Figure 12 histocode

```
[10]: my_matched_src = my_matching_histograms(src,dst)
       compare_matched_hist(src,dst,my_matched_src)
       compare_matched_hist(src,dst,matched_src)
```

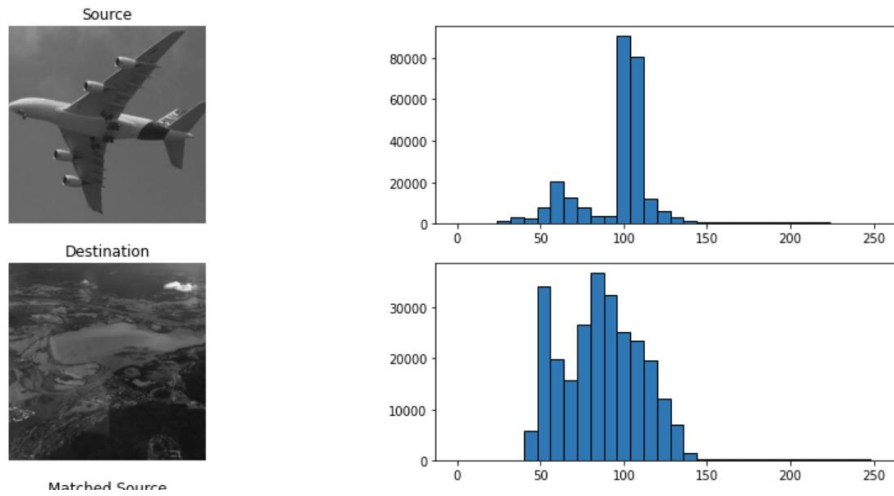
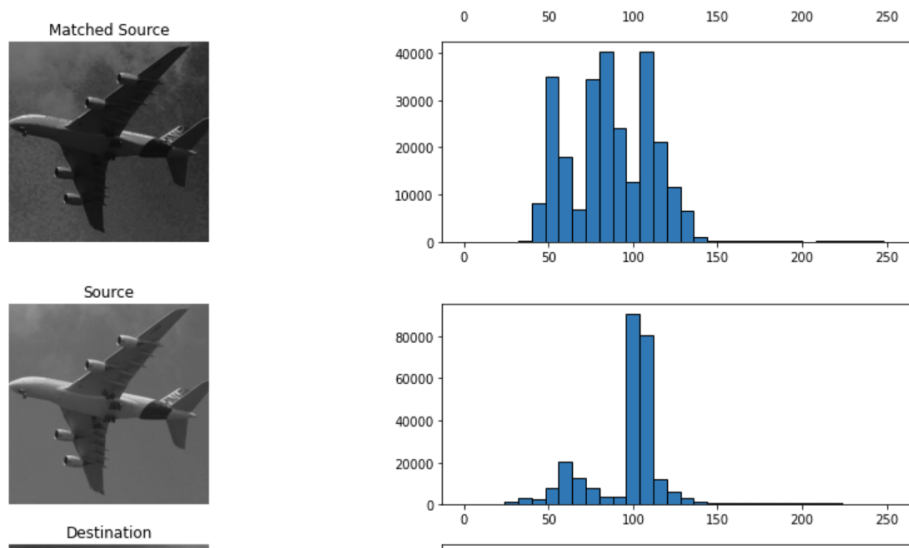
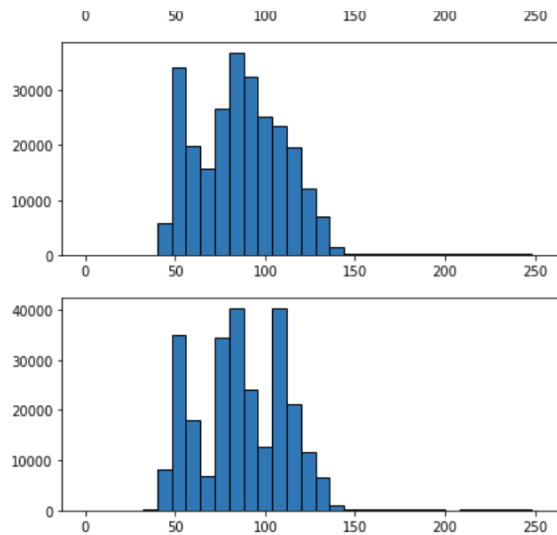


Figure 13 histogram output





Comparing with the inbuilt results, both the image and the histogram appear to be almost similar. The purpose of histogram matching also seems to have been achieved as far as it can be. Since the source and the destination images are completely different one, therefore the matching is not perfect. However the overall lighting and contrast seems to have been matched to the destination image.

```
[11]: src = load_image(1)
      dst = load_image(2)

[12]: %%timeit
      my_equalizing_histograms(src)
      my_matching_histograms(src,dst)

1 loop, best of 5: 203 ms per loop
```

## Reference

- [https://en.wikipedia.org/wiki/Histogram\\_equalization](https://en.wikipedia.org/wiki/Histogram_equalization)
- [https://en.wikipedia.org/wiki/Cumulative\\_distribution\\_function](https://en.wikipedia.org/wiki/Cumulative_distribution_function)
- [https://en.wikipedia.org/wiki/Histogram\\_matching](https://en.wikipedia.org/wiki/Histogram_matching)
- <http://paulbourke.net/miscellaneous/equalisation/>