Project 1

Computer Vision - CS 6643

Out: Sept. 14, 2023
Due: Oct. 5, 2023 (deadline: 11:55PM)

A1) Histogram Equalization

A2) Otsu Image Thresholding

A3) Creative Part: Histogram Equalization for color image

Nidhushan Kanagaraja (N10852881)

A1) Histogram Equalization:

1.1 Introduction

Histogram equalization in general, takes an image and distributes the intensities of the pixels such that we get a more intensity balanced image. We can see this method being implemented to increase the contrast of an image. This method helps us to get more information out of the image, making the image more visually detailed, although this comes with the cost of losing the natural look of the image.

1.2 Probability Density Function (PDF)

The Probability Density Function is the frequency of the intensities of the pixels. We use this to visualize an image, and in this case we are able to see that the intensities are concentrated towards the start of the histogram. This indicates that the image is very dark and has very less white pixels or pixels closer to the white region.

Comparing the images before and after histogram equalization along with their Probability Density Functions (PDFs):

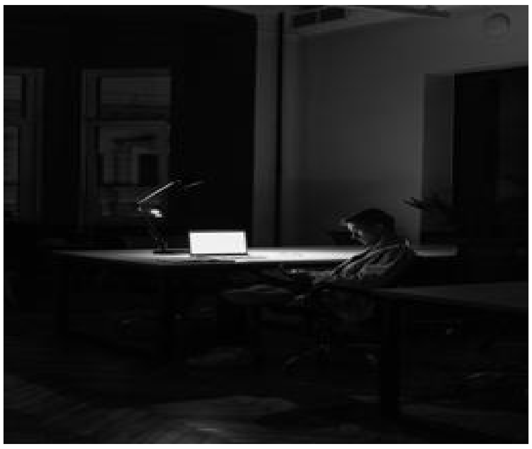

Fig. 1.1 Original Image provided

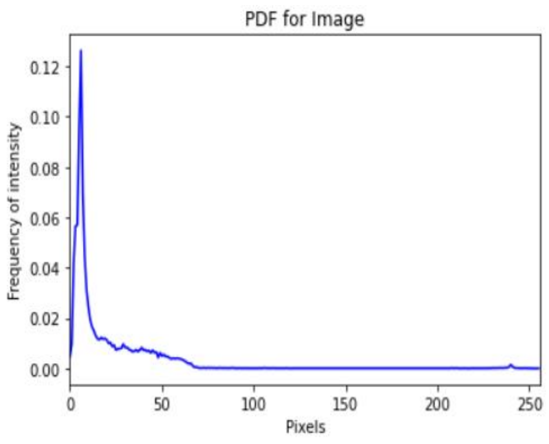

Fig. 1.2 PDF of original image

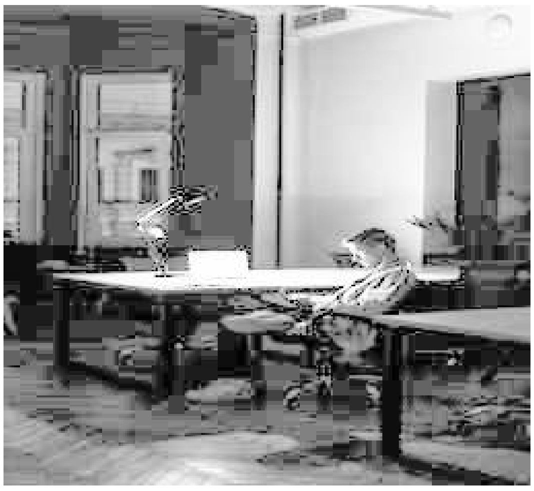

Fig. 1.3 Equalized Image

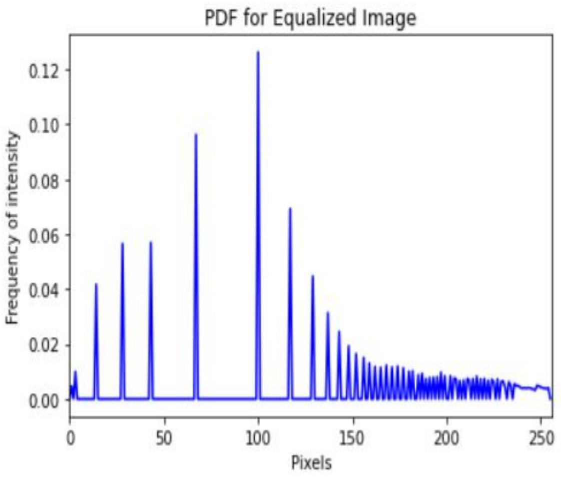

Fig. 1.4 PDF of equalized image

1.3 Observation and Discussion

We can infer from these two images what we have discussed in **1.1 Introduction**. We have an image that is dark and providing us barely any information. Therefore it is almost impossible to make out what is being captured in the image.

After histogram equalization, we can see the image much more clearly and make sense of the whole image. We are able to make out a lot of the things present in the image now. Although there is a loss in the originality and quality of the image, histogram equalization would be helpful in many fields like medical imaging, enhancing images where it is very dark, or has a lot of white regions etc.

Using the histograms provided for each image we can infer that the second histogram is stretching the pixel distribution to utilize more intensity levels, through the whole range. By doing so, we are able to bring the subtle details that are hidden making the image more brighter(in this case).

1.4 Cumulative Distribution Function

Here we compare the Cumulative Distribution Function(CDF) for the original image and the equalized image.

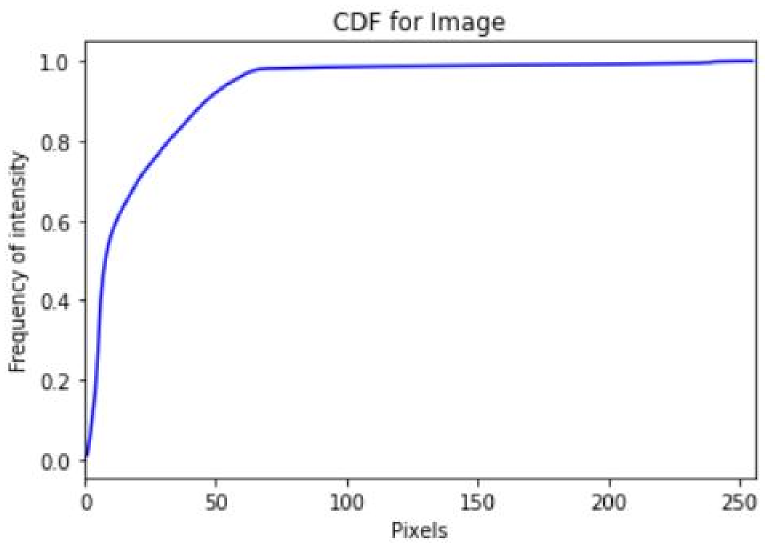

Fig. 1.5 CDF of Original Image

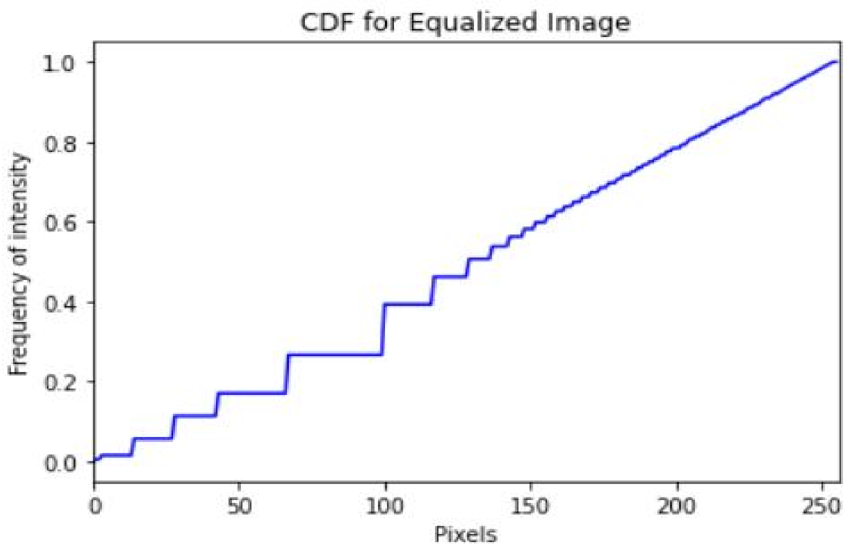

Fig. 1.6 CDF for Equalized Image

Observing the CDF of the original image, it is obvious that the curve is exponentially increasing. The high increase in the start of the curve indicates the high frequency in darker pixels in the image. Whereas in the CDF for the equalized image we are able to see that the increase is spread throughout the values, giving us more pixels on the lighter region. We can observe that this is the case by comparing the original and equalized images.

1.5 Applying Histogram Equalization again

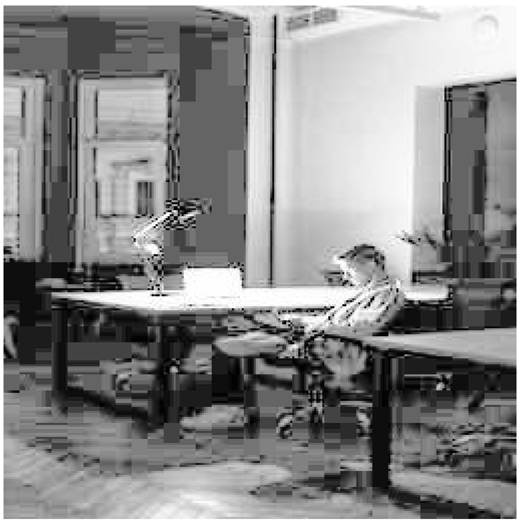

Fig. 1.7 Equalized Image

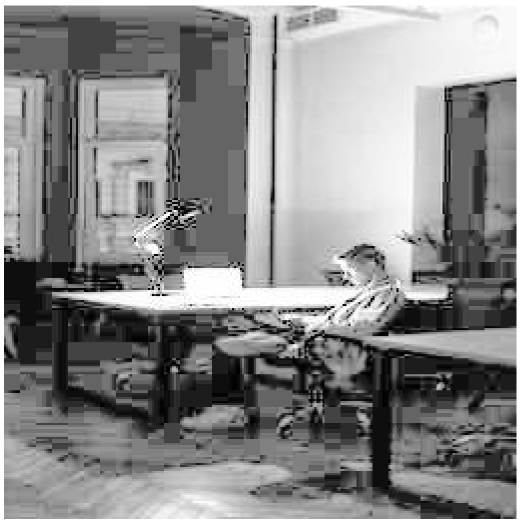

Fig. 1.8 Double Equalized Image

By applying histogram equalization again, we get a very similar result. In all of the graphs observed(PDF, CDF) we see no variation or slight variation. As observed the images look alike and no new information can be obtained from it.

1.6 Testing another image

Taking another sample image taken from my mobile camera, we are able to see similar changes in that image as well. The results are shown below:

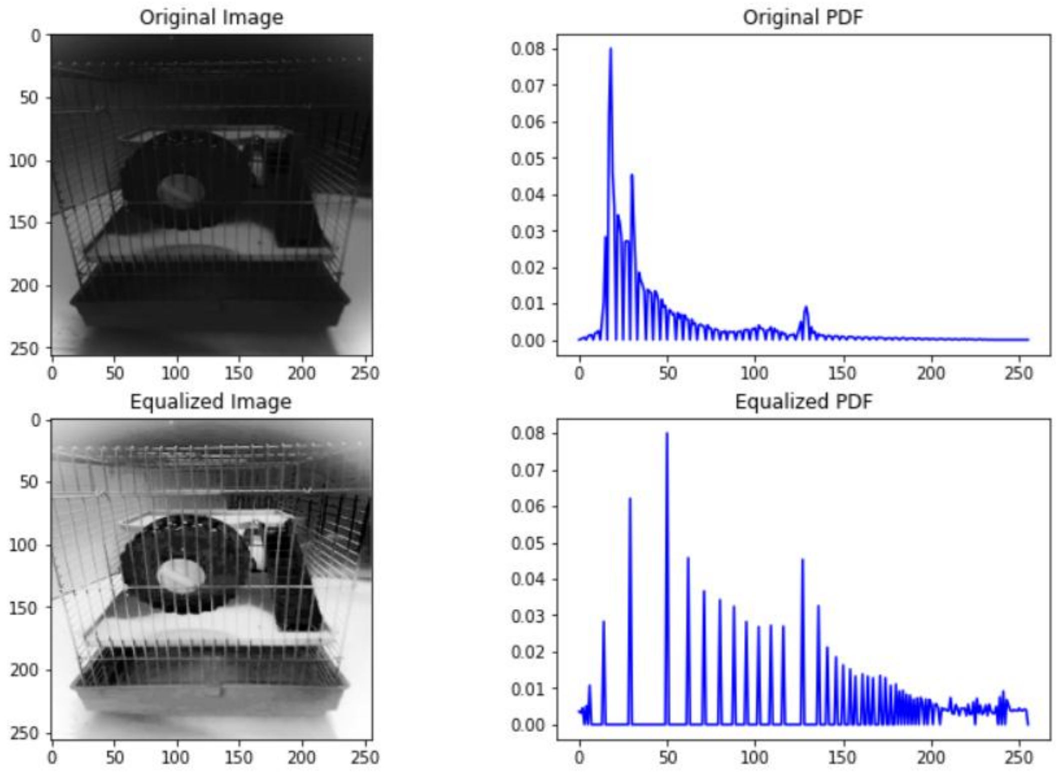

Fig. 1.9 Histogram Equalization for another Image

The equalized image has yet again become more informative and clear. This procedure works well with images having low contrast or high contrast, providing us with better images.

It is again observed that the histogram has been stretched out to the range of values. As for the CDF we have a similar result to our previous image. Both the CDFs, the one from the original image and the one from the equalized image look similar to the CDFs of indoors.png and its equalized version.

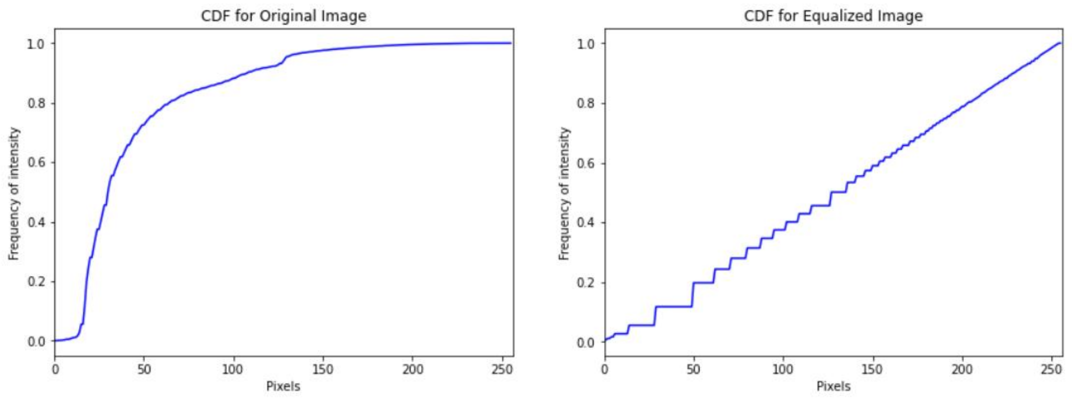

Fig. 1.10 CDF comparison for original and equalized image

A2) Otsu Image Thresholding

2.1 Histograms of each Image:

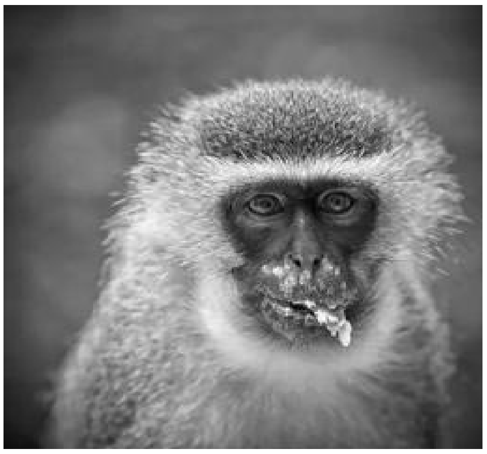
Fig. 2.1 Original Image b2_a.png

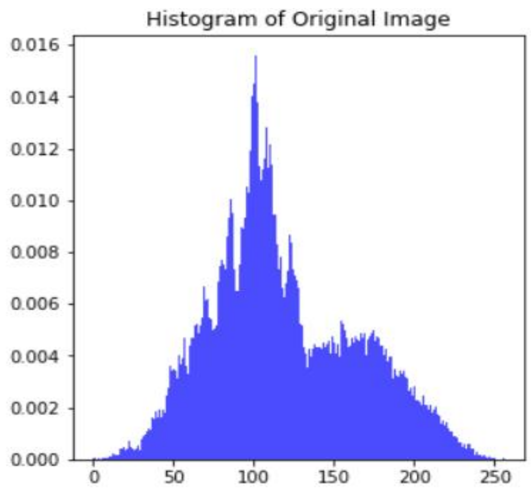

Fig. 2.2 Histogram of Original Image b2_a.png

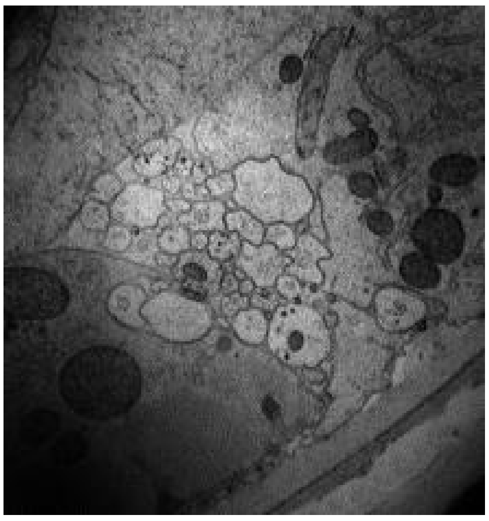
Fig. 2.3 Original Image b2_b.png

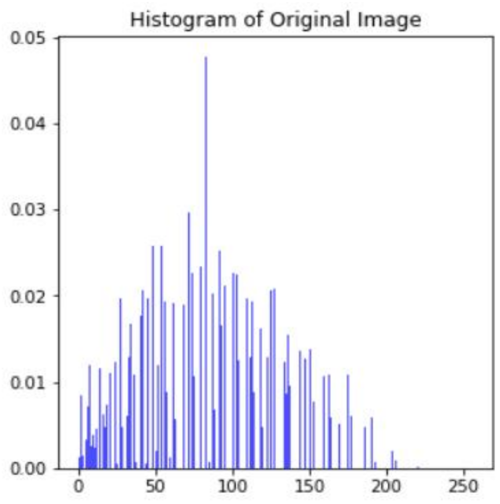

Fig. 2.4 Histogram of Original Image b2_b.png

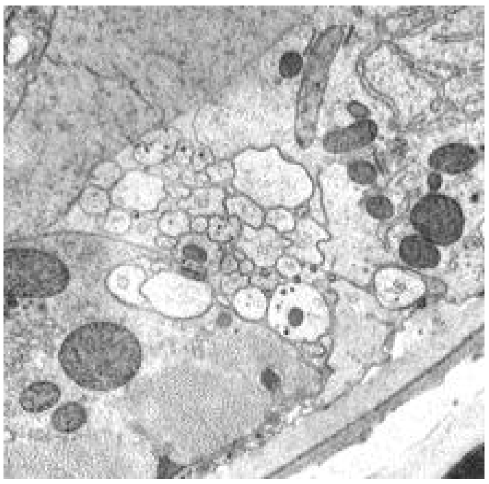
Fig. 2.5 Original Image b2_c.png

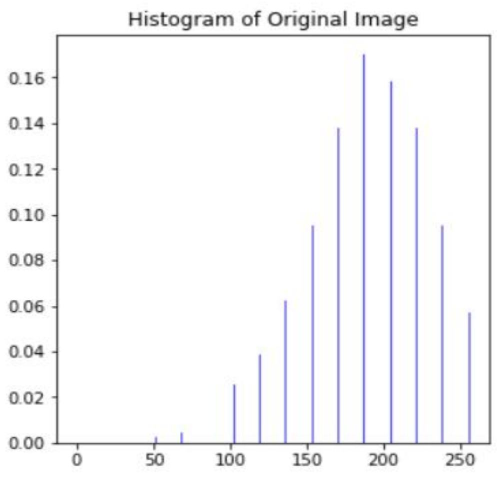

Fig. 2.6 Histogram of Original Image b2_c.png

2.2 Interclass Variance

Now we observe the interclass variance as a function of the chosen threshold.
For image b2_a.png we get the following graph:

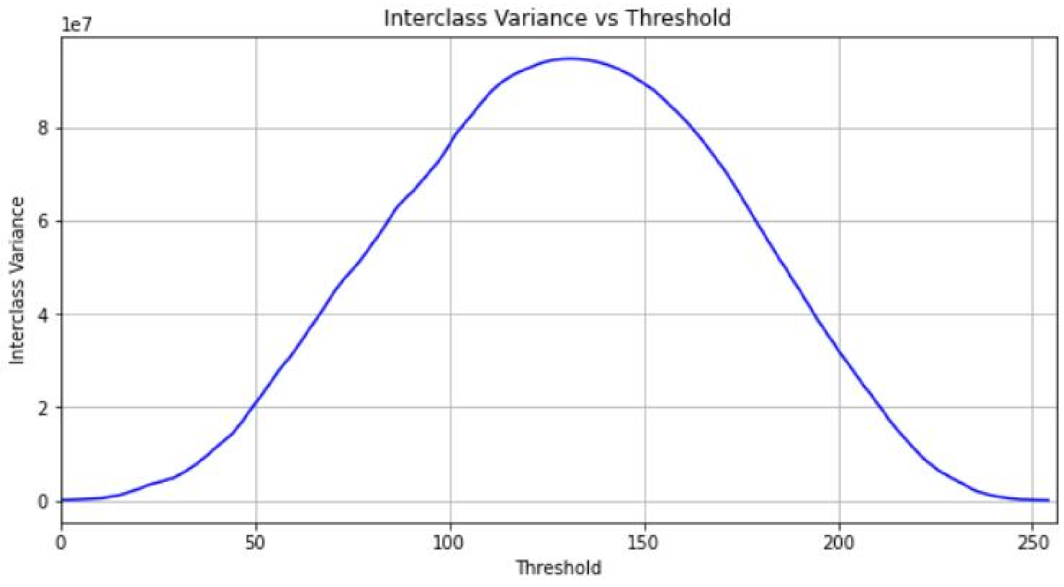

Fig. 2.10 Graph of Interclass Variance Vs Threshold for b2_a.png

```
Otsu Threshold: 131
Otsu Threshold Interclass Variance: 94897019.82898921
Manual Threshold: 150
Max Variance for Manual Threshold: 89609423.30666313
```

For image b2_b.png we get the following graph:

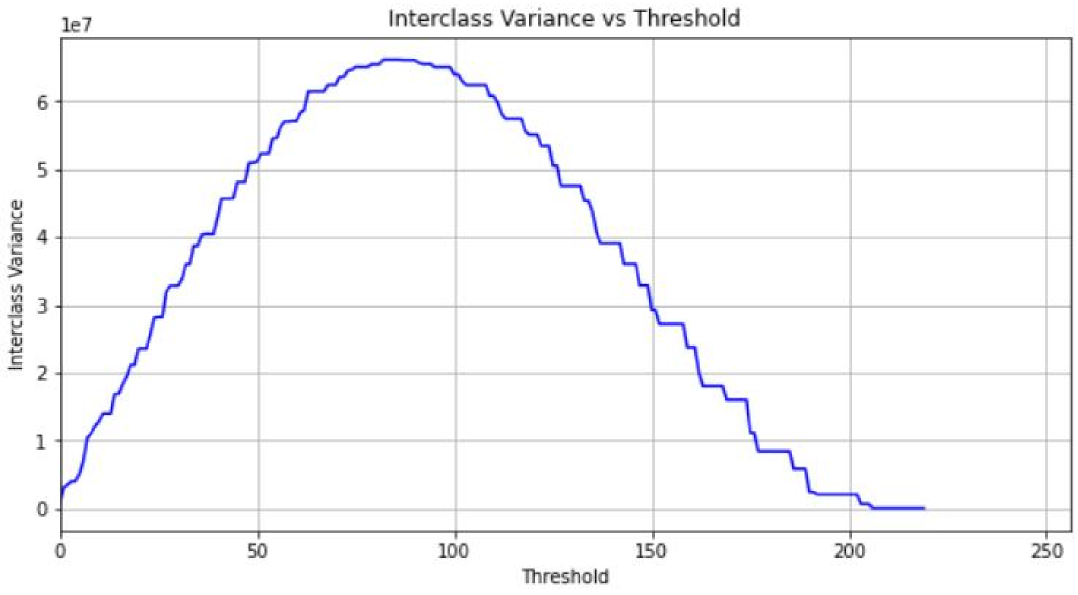

Fig. 2.11 Graph of Interclass Variance Vs Threshold for b2_b.png

```
Otsu Threshold: 85
Otsu Threshold Interclass Variance: 66145754.40544562
Manual Threshold: 100
Max Variance for Manual Threshold: 63999222.37743479
```

For image b2_c.png we get the following graph:

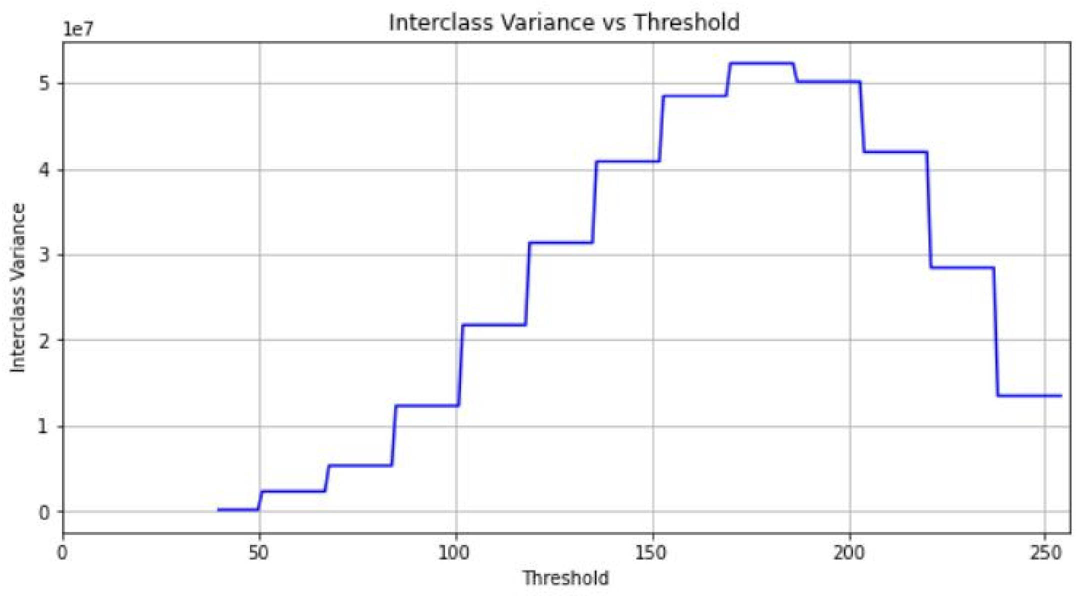

Fig. 2.12 Graph of Interclass Variance Vs Threshold for b2_c.png

```
Otsu Threshold: 170
Otsu Threshold Interclass Variance: 52257901.76515393
Manual Threshold: 175
Max Variance for Manual Threshold: 52257901.76515393
```

More analysis on the Graphs of the interclass variance vs threshold is mentioned in section 2.3 along with the histograms and binary images.

2.3 Observation and Discussion

Now let us observe the effects of both Manual Thresholding and Otsu Thresholding in the histograms of each of the images.
For original image b2_a.png, we get the following result:

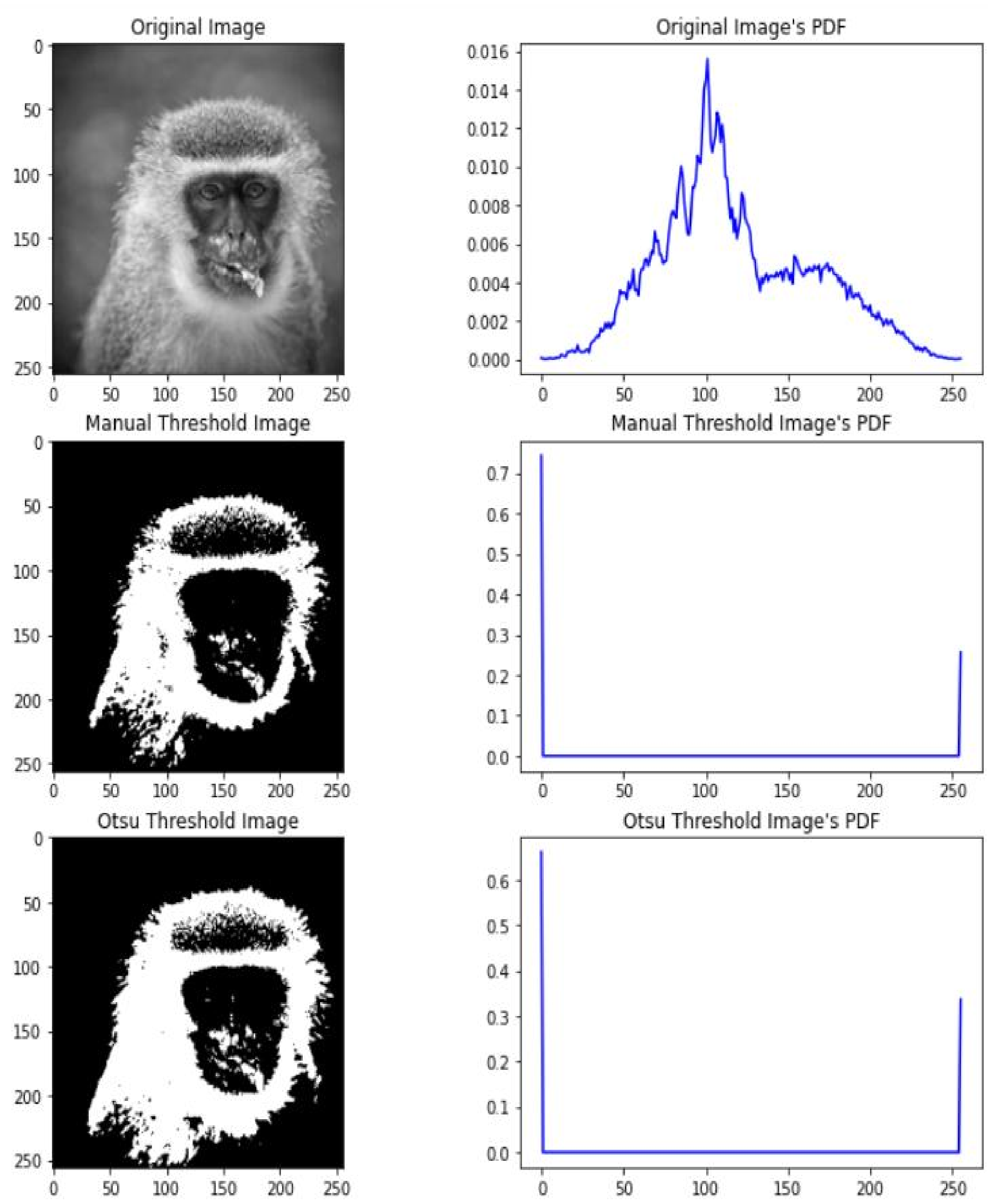

Fig. 2.7 Thresholding comparison for b2_a.png

The Manual Threshold chosen here is 150. As observed the manual threshold is able to single out the hair on the head better than Otsu threshold and Otsu thresholding gives a better outline of the facial structure and the body to the bottom left. The histogram is not skewed on either sides for this image and it has continuous values. We will see why this matters in the next 2 images.

For image b2_b.png, we get the following result:

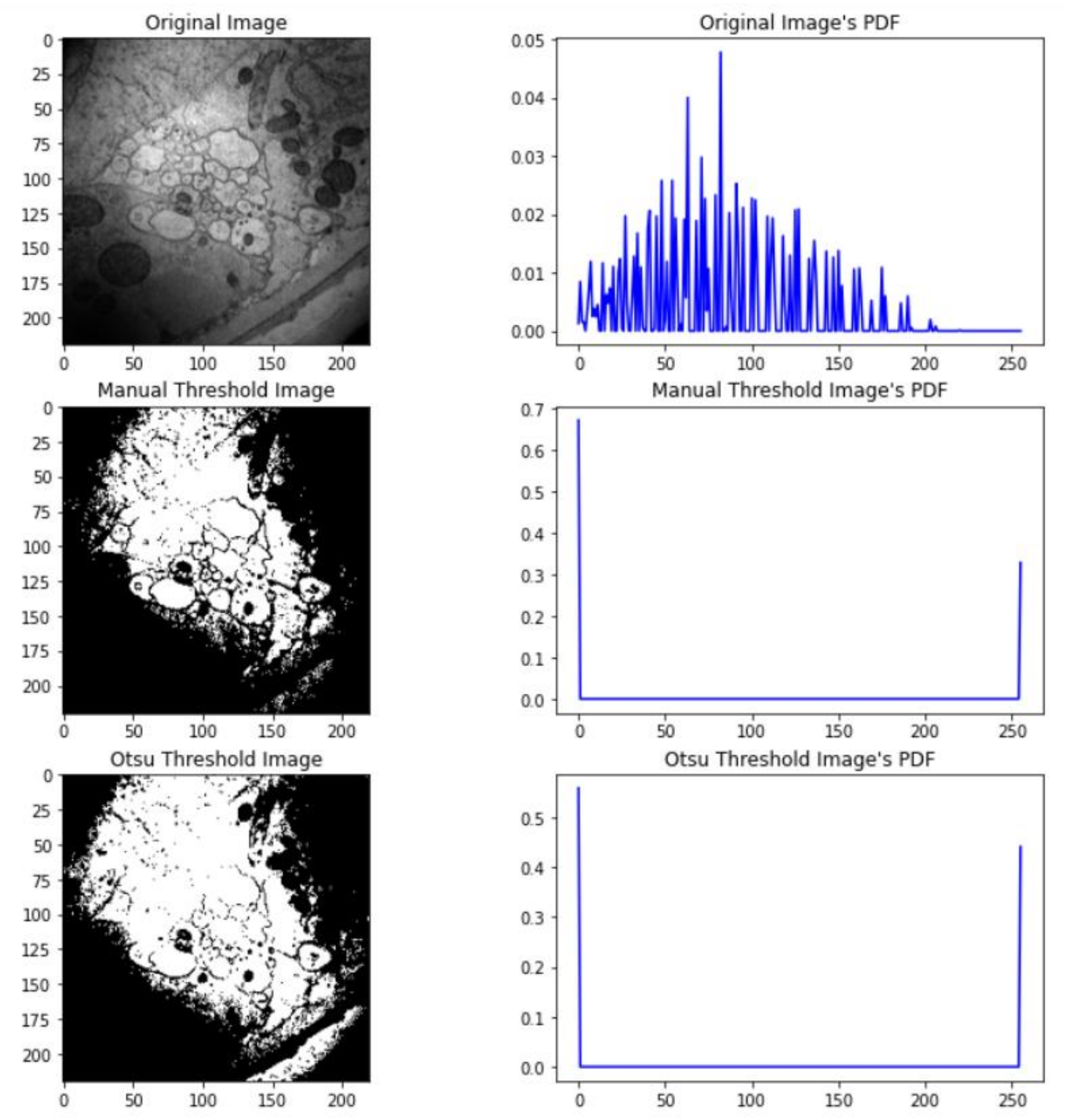

Fig. 2.8 Thresholding comparison for b2_b.png

Here, a manual threshold of 100 has been used. In the manual thresholding the membrane to the bottom right seems to be less visible but the center part looks better than Otsu thresholding.

Here the histogram is skewed to the left side. This is important as we can observe from section 2.2, that the interclass variance has also been skewed to the left for this image. Therefore, just using the histogram we will be able to find that the optimal threshold will be closer to 0.

Now moving on to continuity, as this histogram is not continuous, the interclass variance also has small ups and downs, not forming a smooth continuous line.

For image b2_c.png, we get the following result:

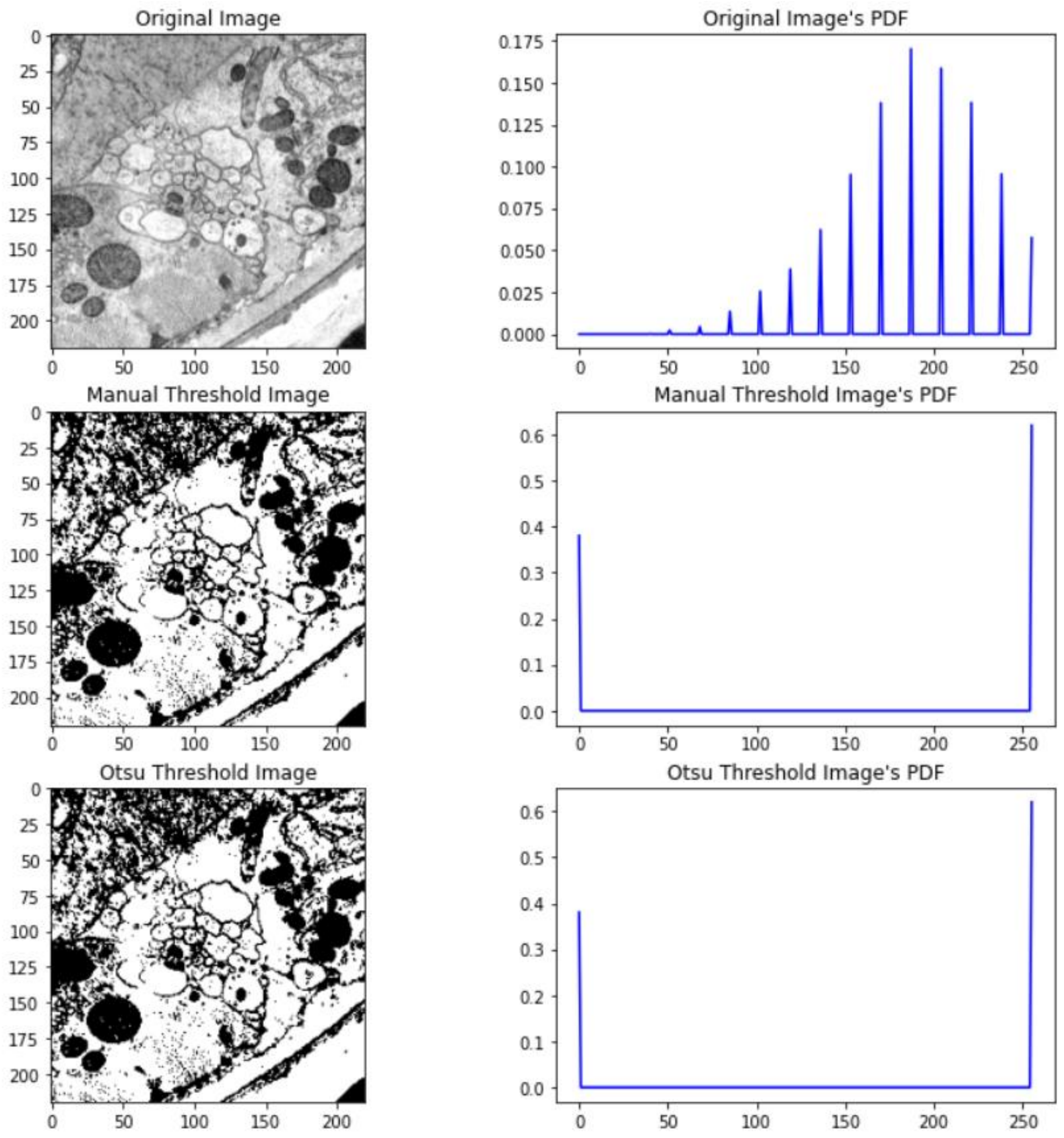

Fig. 2.9 Thresholding comparison for b2_c.png

Here a manual threshold of 175 is used. As the Otsu threshold(170) is very close to the manual threshold chosen, there is barely any difference in the binary images. In fact, as observed in Section 2.2, we see that the interclass variance is the same for both thresholds.

Furthermore, we also see here that the histogram has been skewed to the right in this case. So, similar to the previous case, here also we have a graph of interclass variance that has been skewed to the right. As for continuity, there are larger breaks in the values of the histogram, and its effects are clearly visible in the plot of the interclass variance. We see larger ups and downs on the graph.

2.3.1 Comparison between manual and Otsu Threshold

Otsu threshold has a high advantage when it comes to real world scenario's as people don't need any image specific information or expertise when handling it. It is also helpful when we don't know the dataset or have to deal with different images in different lighting conditions.

Manual Thresholding involves us in selecting a threshold for the image based on our knowledge. This gives us a better chance of getting our desired result from the input image. For instance, in image b2_a.png, if my requirement was to focus on the hair on the head, manual thresholding has an upper handing. Similarly in image b2_b.png, depending on whether the membrane is important or center part of the image is important, both techniques have an advantage over the other.

The Automatic threshold does produce an above average result each time, without any human interference. Yes, the results could be improved if we had specific requirements as mentioned above for manual thresholding, but it would require some knowledge on both the image and manual thresholding to get a good result.

A3) Creative Part

3. Histogram Equalization for a color image

Original Image:

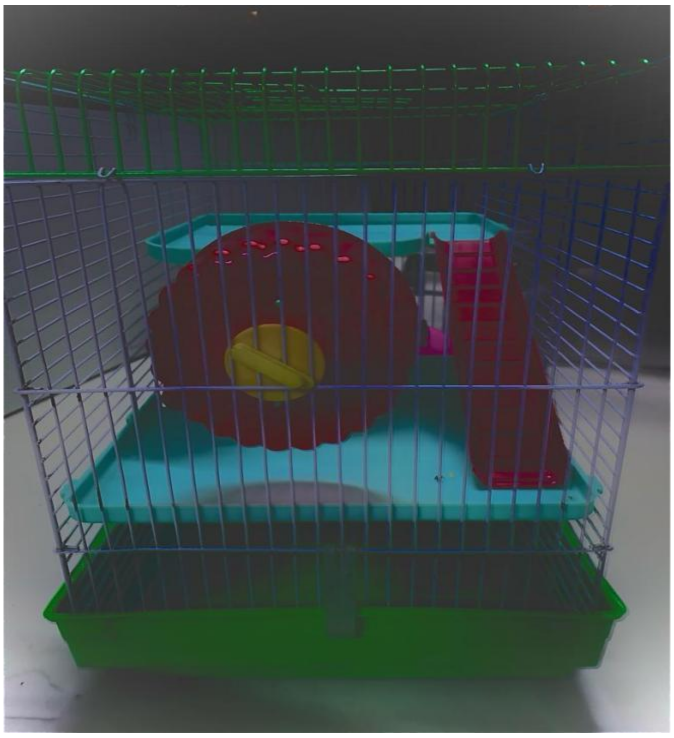
Fig. 3.1 Original color image

Equalized Image:

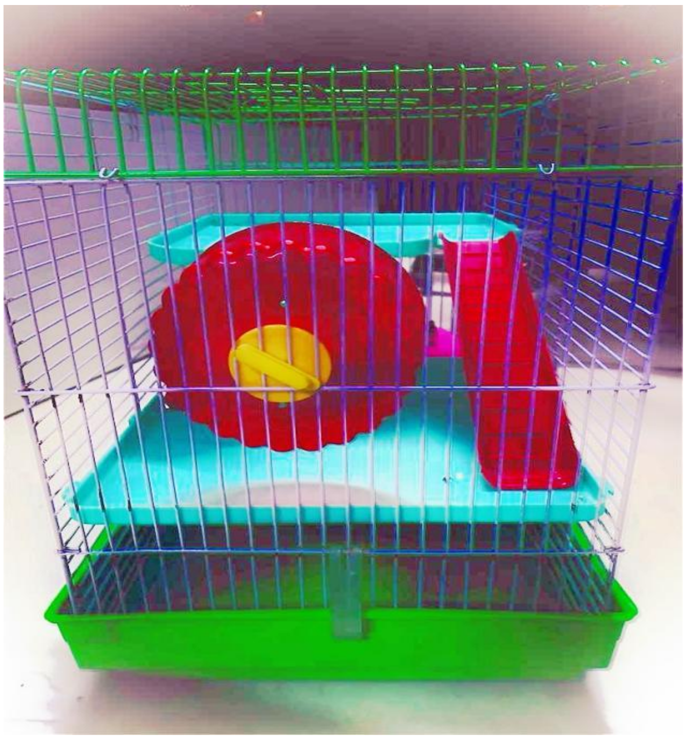
Fig. 3.2 Equalized color image

For color images, the process is very similar to grayscale histogram equalization. The image is split into three parts, the red, blue and greens in the image. Now histogram equalization is done to all three of them, and in the end the image is merged.

3.1 Comparison of Histograms

Let us compare the PDFs and CDFs of all three options, blue, green, and red for the original and equalized versions.

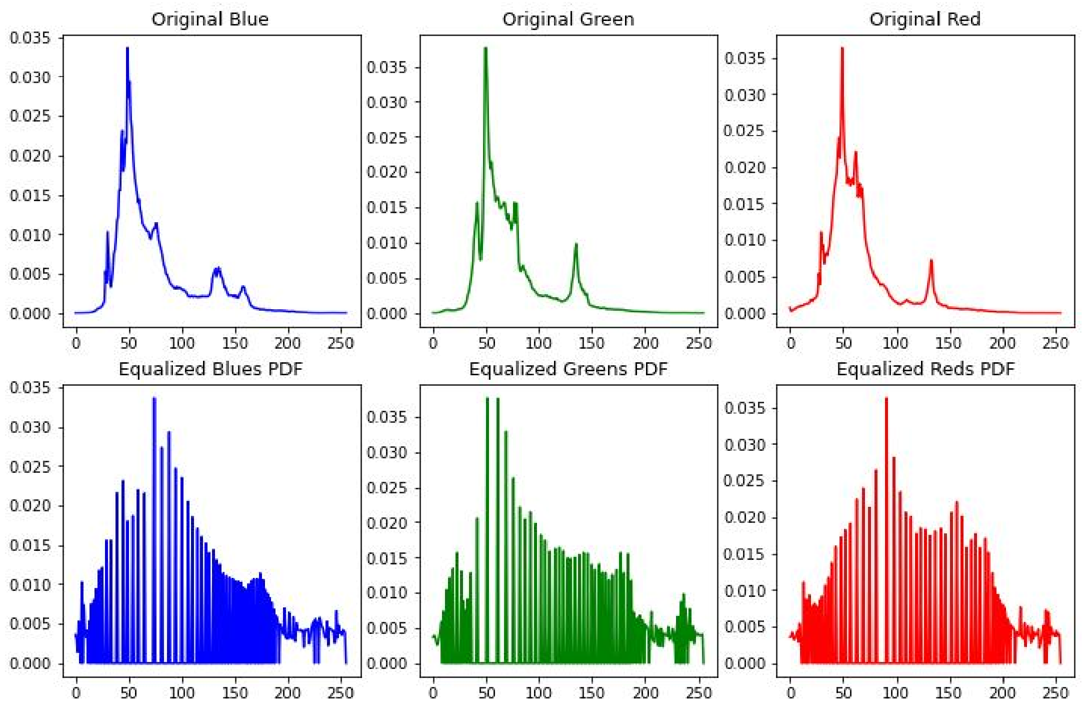

Fig. 3.3 Comparison of PDFs for the three different colors for equalized and original images

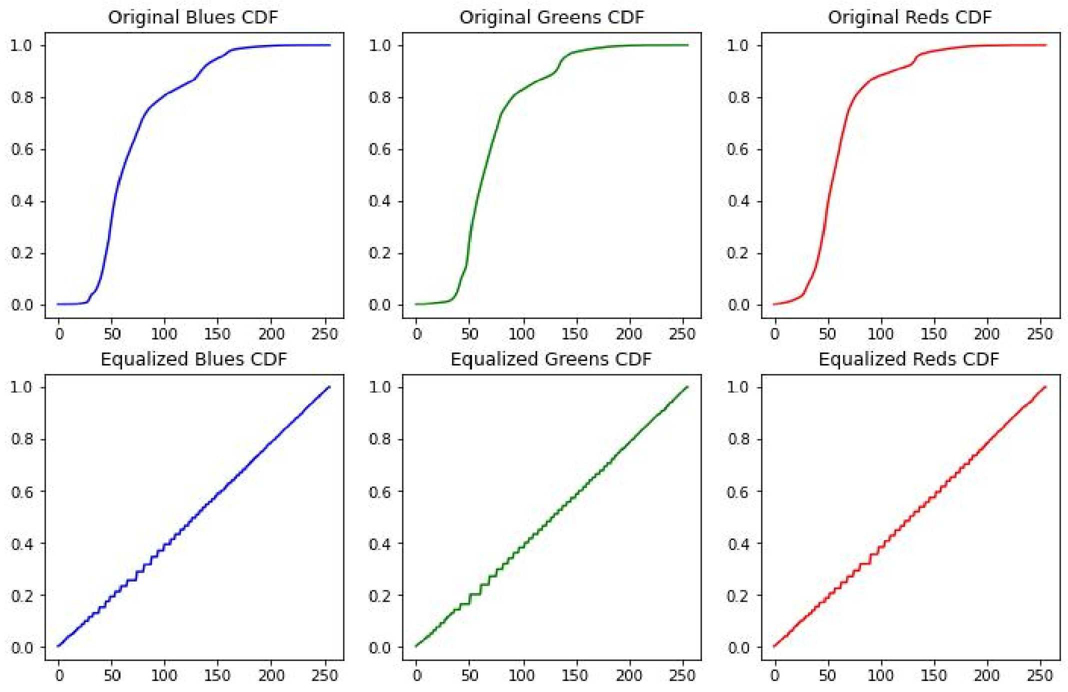

Fig. 3.4 Comparison of CDFs for the three different colors for equalized and original images

Upon observation, the three have minor differences, and we get similar results while we compute PDFs and CDFs for the three as we saw in section 1. The PDFs have stretched throughout the values.

The CDF is a smooth increasing line for the equalized image as expected. The main observation here is that the three different colors have hardly any difference in their graphs. This will need further investigation to conclude if that is the case for all images, or will it change for images where one shade of the color is more.

Appendix

Libraries Required:

In [ ]:
```
import cv2
import numpy as np
import matplotlib.pyplot as plt
from PIL import Image
```

A1) Histogram Equalization

In [ ]:
```
image = cv2.imread('indoors.png', cv2.IMREAD_GRAYSCALE)
```

In [ ]:
```
original_image = Image.open('indoors.png')
resized_image = original_image.resize((256, 256))
im = np.array(resized_image.convert('L'))
image = ((im - np.min(im)) * (1/(np.max(im) - np.min(im)) * 255)).astype('u
int8')
```

In [ ]:
```
def create_pdf(im_in):
    # Create normalized intensity histogram from an input image
    unNorm_hist = [0] * 256
    number_of_pixels=0
    for row in im_in:
        for pixel in row:
            unNorm_hist[pixel] += 1
            number_of_pixels+=1
    pdf = [(freq / number_of_pixels) for freq in unNorm_hist]
    return pdf
```

In [ ]:
```
def create_cdf(pdf):
    # Create the cumulative distribution function from an input pdf
    cdf = [sum(pdf[:i+1]) for i in range(len(pdf))]
    return cdf
```

In [ ]:
```
def histogram_equalization(im_in):
    pdf = create_pdf(im_in)
    cdf = create_cdf(pdf)
    equalized_im = np.zeros_like(im_in)

    for i in range(im_in.shape[0]):
        for j in range(im_in.shape[1]):
            pixel_value = im_in[i, j]
            equalized_pixel = int(255 * cdf[pixel_value])
            equalized_im[i, j] = equalized_pixel
    # Create a histogram equalized image using your computed cdf
    return equalized_im
```

In [ ]:
```python
pdf = create_pdf(image)

plt.plot(pdf, color='b')
plt.xlim([0, 256])
plt.title('PDF for Image')
plt.xlabel('Pixels')
plt.ylabel('Frequency of intensity')
plt.show()
```

In [ ]:
```python
cdf = create_cdf(pdf)

plt.plot(cdf, color='b')
plt.xlim([0, 256])
plt.title('CDF for Image')
plt.xlabel('Pixels')
plt.ylabel('Frequency of intensity')
plt.show()
```

In [ ]:
```python
eq_image = histogram_equalization(image)

eq_pdf = create_pdf(eq_image)

# cv2.imwrite("personal2_Q1_eq.png", eq_image)

plt.figure(figsize=(12, 8))
plt.subplot(2, 2, 1)
plt.imshow(image, cmap='gray')
plt.title('Original Image')

plt.subplot(2, 2, 2)
plt.plot(pdf, color='b')
plt.title('Original PDF')

plt.subplot(2, 2, 3)
plt.imshow(eq_image, cmap='gray')
plt.title('Equalized Image')

plt.subplot(2, 2, 4)
plt.plot(eq_pdf, color='b')
plt.title('Equalized PDF')

plt.show()
```

In [ ]:
```python
plt.plot(eq_pdf, color='b')
plt.xlim([0, 256])
plt.title('PDF for Equalized Image')
plt.xlabel('Pixels')
plt.ylabel('Frequency of intensity')
plt.show()
```

In [ ]:
```python
eq_cdf = create_cdf(eq_pdf)
plt.plot(eq_cdf, color='b')
plt.xlim([0, 256])
plt.title('CDF for Equalized Image')
plt.xlabel('Pixels')
plt.ylabel('Frequency of intensity')
plt.show()
```

In [ ]:
```
cv2.imshow('Original Image', image)
cv2.imshow('Equalized Image', eq_image)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

A2) Otsu Image Thresholding

In [ ]:
```
# READING THE THREE IMAGES
b2a = cv2.imread('b2_a.png', cv2.IMREAD_GRAYSCALE)
b2b = cv2.imread('b2_b.png', cv2.IMREAD_GRAYSCALE)
b2c = cv2.imread('b2_c.png', cv2.IMREAD_GRAYSCALE)
```

In [ ]:
```
def manual_threshold(im_in, threshold):
    # Threshold image with the threshold of your choice
    manual_thresh_img = np.where(im_in > threshold, 255, 0).astype(np.uint
8)
    return manual_thresh_img
```

In [ ]:
```
def otsu_threshold(im_in):
    # Create Otsu thresholded image
    max_variance = 0
    otsu_threshold = 0
    for i in range(256):
        interclass_variance = calc_int_class_var(image,i)
        if interclass_variance > max_variance:
            max_variance = interclass_variance
            otsu_threshold = i
    otsu_thresh_img = np.where(im_in > otsu_threshold, 255, 0).astype(np.ui
nt8)
    print("Otsu Threshold:",otsu_threshold)
    print("Max Variance:",max_variance)
    return otsu_thresh_img
```

In [ ]:
```
def calc_int_class_var(image,threshold):
    background = image[image <= threshold]
    foreground = image[image > threshold]

    wb = len(background) / len(image)
    wf = len(foreground) / len(image)

    mean_b = np.mean(background)
    mean_f = np.mean(foreground)

    inter_class_variance = wb * wf * (mean_b - mean_f) ** 2

    return inter_class_variance
```

In [ ]:
```python
image = cv2.imread('b2_a.png', cv2.IMREAD_GRAYSCALE)

pdf_orig = create_pdf(image)

man_threshold = 150
man_thresh_img = manual_threshold(image, man_threshold)
pdf_man = create_pdf(man_thresh_img)

otsu_thresh_img = otsu_threshold(image)
pdf_otsu = create_pdf(otsu_thresh_img)

plt.figure(figsize=(12, 12))
plt.subplot(3, 2, 1)
plt.imshow(image, cmap='gray')
plt.title('Original Image')

plt.subplot(3, 2, 2)
plt.plot(pdf_orig, color='b')
plt.title('Original Image\'s PDF')

plt.subplot(3, 2, 3)
plt.imshow(man_thresh_img, cmap='gray')
plt.title('Manual Threshold Image')

plt.subplot(3, 2, 4)
plt.plot(pdf_man, color='b')
plt.title('Manual Threshold Image\'s PDF')

plt.subplot(3, 2, 5)
plt.imshow(otsu_thresh_img, cmap='gray')
plt.title('Otsu Threshold Image')

plt.subplot(3, 2, 6)
plt.plot(pdf_otsu, color='b')
plt.title('Otsu Threshold Image\'s PDF')

plt.show()

print("Manual Threshold:",man_threshold)
print("Max Variance for Manual Threshold:", calc_int_class_var(image,man_th
reshold))
cv2.imshow('Manual Thresholding', man_thresh_img)
cv2.imshow('Otsu Thresholding', otsu_thresh_img)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

In [ ]:
```python
interclass_variance_list = [calc_int_class_var(image,i) for i in range(25
6)]
# print(interclass_variance_list)
plt.figure(figsize=(10, 5))
plt.plot(interclass_variance_list, color='b')
plt.xlim([0, 256])
plt.title('Interclass Variance vs Threshold')
plt.xlabel('Threshold')
plt.ylabel('Interclass Variance')
plt.grid(True)
plt.show()
```

In [ ]:

A3) Creative Part

Making Histogram Equalization for colour images

In [ ]:
```
image = cv2.imread("personal2_Q1_color.jpg")
cv2.imshow('orig img', image)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

In [ ]:
```
blue, green, red = cv2.split(image)
cv2.imshow('blue',blue)
cv2.imshow('green',green)
cv2.imshow('red',red)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

In [ ]:
```
eq_blue = histogram_equalization(blue)
eq_green = histogram_equalization(green)
eq_red = histogram_equalization(red)
```

In [ ]:
```
equalized_image = cv2.merge((eq_blue, eq_green, eq_red))
```

In [ ]:
```
cv2.imwrite("personal2_Q1_color_eq.jpg", equalized_image)

# Display the original and equalized images using OpenCV
cv2.imshow("Original Image", image)
cv2.imshow("Equalized Image", equalized_image)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

In [ ]:
```
blue_pdf =  create_pdf(blue)
green_pdf = create_pdf(green)
red_pdf = create_pdf(red)

eq_blue_pdf = create_pdf(eq_blue)
eq_green_pdf = create_pdf(eq_green)
eq_red_pdf = create_pdf(eq_red)

plt.figure(figsize=(12, 8))

plt.subplot(2, 3, 1)
plt.plot(blue_pdf, color='b')
plt.title('Original Blue')

plt.subplot(2, 3, 2)
plt.plot(green_pdf, color='g')
plt.title('Original Green')

plt.subplot(2, 3, 3)
plt.plot(red_pdf, color='r')
plt.title('Original Red')

plt.subplot(2, 3, 4)
plt.plot(eq_blue_pdf, color='b')
plt.title('Equalized Blues PDF')

plt.subplot(2, 3, 5)
plt.plot(eq_green_pdf, color='g')
plt.title('Equalized Greens PDF')

plt.subplot(2, 3, 6)
plt.plot(eq_red_pdf, color='r')
plt.title('Equalized Reds PDF')


plt.show()
```

In [ ]:
```
blue_cdf =  create_cdf(blue_pdf)
green_cdf = create_cdf(green_pdf)
red_cdf = create_cdf(red_pdf)

eq_blue_cdf =  create_cdf(eq_blue_pdf)
eq_green_cdf = create_cdf(eq_green_pdf)
eq_red_cdf = create_cdf(eq_red_pdf)

plt.figure(figsize=(12, 8))

plt.subplot(2, 3, 1)
plt.plot(blue_cdf, color='b')
plt.title('Original Blues CDF')

plt.subplot(2, 3, 2)
plt.plot(green_cdf, color='g')
plt.title('Original Greens CDF')

plt.subplot(2, 3, 3)
plt.plot(red_cdf, color='r')
plt.title('Original Reds CDF')

plt.subplot(2, 3, 4)
plt.plot(eq_blue_cdf, color='b')
plt.title('Equalized Blues CDF')

plt.subplot(2, 3, 5)
plt.plot(eq_green_cdf, color='g')
plt.title('Equalized Greens CDF')

plt.subplot(2, 3, 6)
plt.plot(eq_red_cdf, color='r')
plt.title('Equalized Reds CDF')

plt.show()
```

In [ ]: