# PCV Lab-7

190030485

G G Sanjana

## Prelab

1. What is an image histogram and why it is useful?

A. An image histogram is a type of histogram that acts as a graphical representation of the tonal distribution in a digital image. It plots the number of pixels for each tonal value. By looking at the histogram for a specific image a viewer will be able to judge the entire tonal distribution at a glance.

Image histograms are present on many modern digital cameras. Photographers can use them as an aid to show the distribution of tones captured, and whether image detail has been lost to blown-out highlights or blacked-out shadows. This is less useful when using a raw image format, as the dynamic range of the displayed image may only be an approximation to that in the raw file.

The horizontal axis of the graph represents the tonal variations, while the vertical axis represents the total number of pixels in that particular tone.

The left side of the horizontal axis represents the dark areas, the middle represents mid-tone values, and the right-hand side represents light areas. The vertical axis represents the size of the area (total number of pixels) that is captured in each one of these zones.

Thus, the histogram for a very dark image will have most of its data points on the left side and centre of the graph.

2. What is histogram equalization?

A. Histogram Equalization is a computer image processing technique used to improve contrast in images. It accomplishes this by effectively spreading out the most frequent intensity values, i.e. stretching out the intensity range of the image. This method usually increases the global contrast of images when its usable data is represented by close contrast values. This allows for areas of lower local contrast to gain a higher contrast.

A color histogram of an image represents the number of pixels in each type of color component. Histogram equalization cannot be applied separately to the Red, Green and Blue components of the image as it leads to dramatic changes in the image's color balance. However, if the image is first converted to another color space, like HSL/HSV color space, then the algorithm can be applied to the luminance or value channel without resulting in changes to the hue and saturation of the image.

3. Write the pseudo-code for histogram equalization.

A.

```
// Pseudo-code to perform histogram equalisation to adjust contrast levels
// Function to perform histogram equalisation on an image
// Function takes total rows, columns, input file name and output
// file name as parameters
void histogramEqualisation(int cols, int rows,
                                             char* input_file_name, char*
output_file_name)
{
    // creating image pointer
    unsigned char* image;
    // Declaring 2 arrays for storing histogram values (frequencies) and
    // new gray level values (newly mapped pixel values as per algorithm)
    int hist[256] = { 0 };
    int new_gray_level[256] = { 0 };
    // Declaring other important variables
    int input_file, output_file, col, row, total, curr, i;
    // allocating image array the size equivalent to number of columns
    // of the image to read one row of an image at a time
    image = (unsigned char*)calloc(cols, sizeof(unsigned char));
    // opening input file in Read Only Mode
```

```
input_file = open(input_file_name, O_RDONLY);
if (input_file < 0) {
        printf("Error opening input file\n");
        exit(1);
}
// creating output file that has write and read access
output_file = creat(output_file_name, 0666);
if (output_file < 0) {
        printf("Error creating output file\n");
        exit(1);
}
// Calculating frequency of occurrence for all pixel values
for (row = 0; row < rows; row++) {
        // reading a row of image
        read(input_file, &image[0], cols * sizeof(unsigned char));
        // logic for calculating histogram
        for (col = 0; col < cols; col++)
                hist[(int)image[col]]++;
}
// calculating total number of pixels
total = cols * rows;
curr = 0;
// calculating cumulative frequency and new gray levels
for (i = 0; i < 256; i++) {
        // cumulative frequency
        curr += hist[i];
```

```
            // calculating new gray level after multiplying by
            // maximum gray count which is 255 and dividing by
            // total number of pixels
            new_gray_level[i] = round((((float)curr) * 255) / total);
        }
        // closing file
        close(input_file);
        // reopening file in Read Only Mode
        input_file = open(input_file_name, O_RDONLY);
        // performing histogram equalisation by mapping new gray levels
        for (row = 0; row < rows; row++) {
            // reading a row of image
            read(input_file, &image[0], cols * sizeof(unsigned char));
            // mapping to new gray level values
            for (col = 0; col < cols; col++)
                image[col] = (unsigned char)new_gray_level[image[col]];
            // reading new gray level mapped row of image
            write(output_file, &image[0], cols * sizeof(unsigned char));
        }
    }
    // driver code
    main()
    {
        // declaring variables
        char* input_file_name;
        char* output_file_name;
        int cols, rows;
        // defining number of rows and columns in an image
        // here, image size is 512*512
        cols = 512;
        rows = 512;
        // defining input file name (input image name)
        // this boat_512_512 is a raw grayscale image
        input_file_name = "boat_512_512";
        // defining output file name (output image name)
        output_file_name = "boat_512_512_histogram_equalised";
        // calling function to do histogram equalisation
        histogramEqualisation(cols, rows, input_file_name, output_file_name);
    }
```

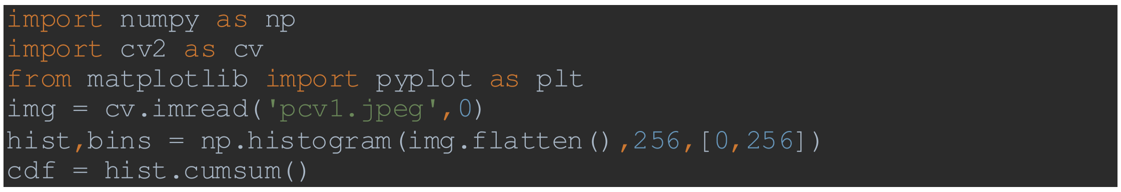
4. Discuss some applications of histogram equalization.

A. Histogram equalization is a straightforward image-processing technique often used to achieve better quality images in black and white colour scales in medical applications such as digital X-rays, MRIs, and CT scans. All these images require high definition and contrast of colours to determine the pathology that is being observed and reach a diagnosis. However, in some type of images histogram equalization can show noise hidden in the image after the processing is done. Therefore it is often used with other imaging processing techniques.

## INLAB

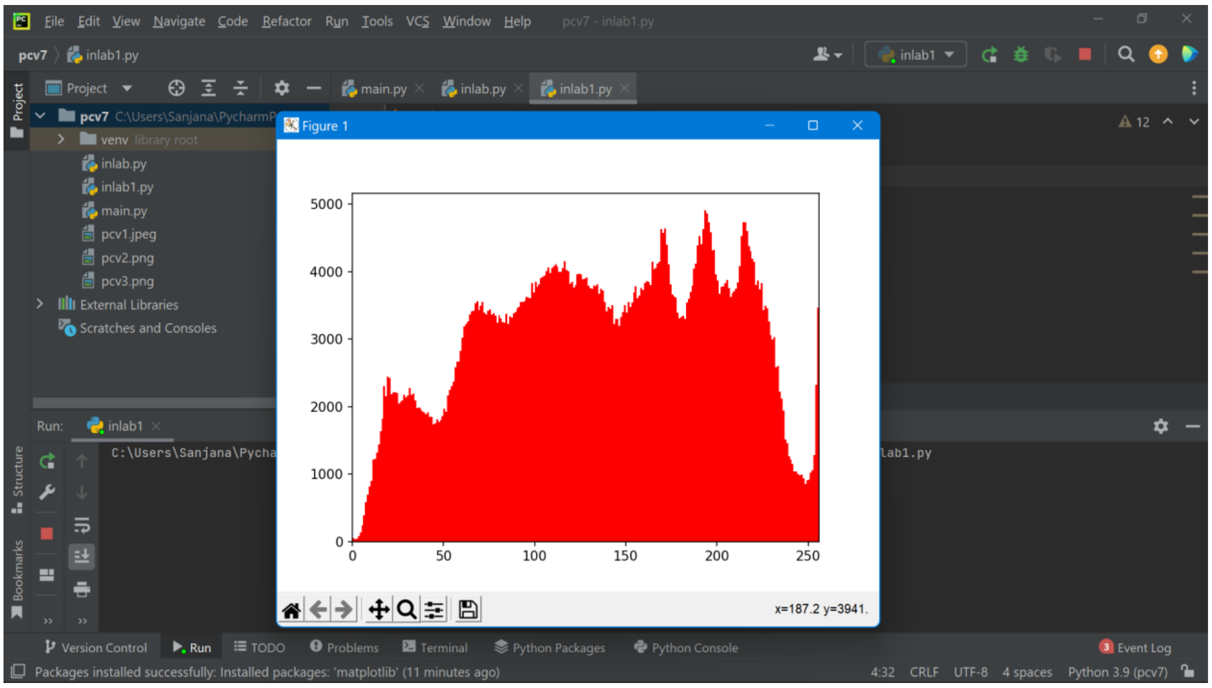
1. Apply histogram equalization using the equalizeHist function.

A.



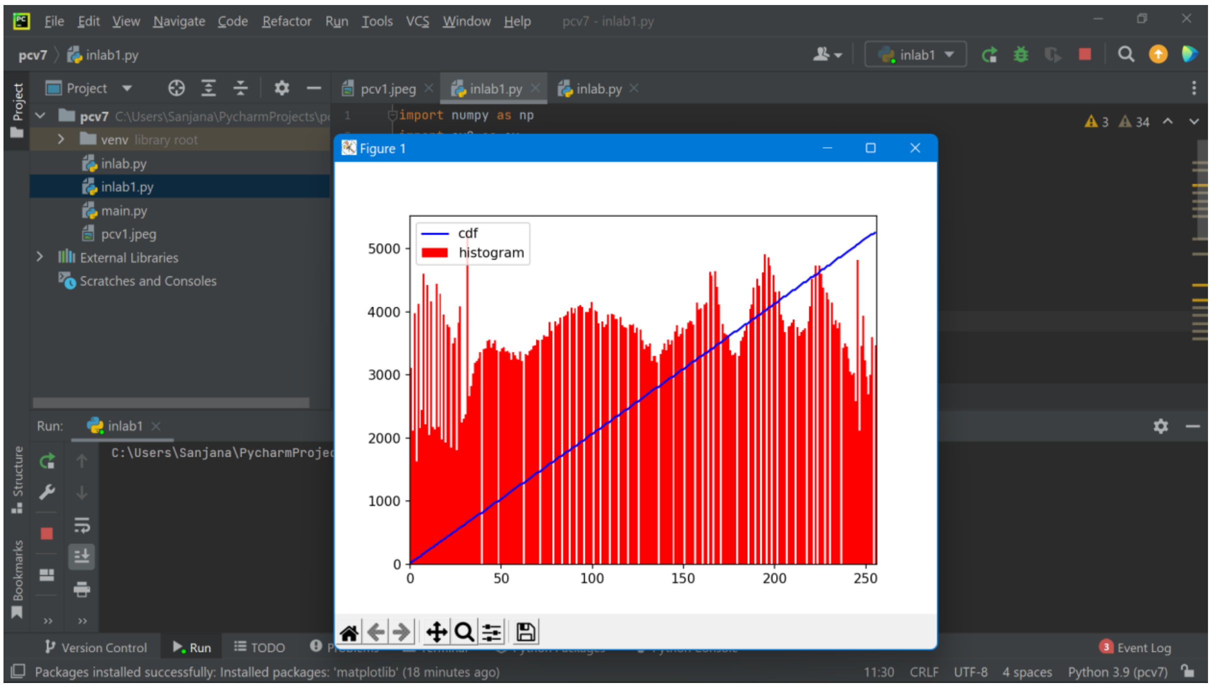
```
import numpy as np
import cv2 as cv
from matplotlib import pyplot as plt
img = cv.imread('pcv1.jpeg',0)
hist,bins = np.histogram(img.flatten(),256,[0,256])
cdf = hist.cumsum()
```

```
cdf_normalized = cdf * float(hist.max()) / cdf.max()
plt.plot(cdf_normalized, color = 'b')
plt.hist(img.flatten(),256,[0,256], color = 'r')
plt.xlim([0,256])
plt.legend(('cdf','histogram'), loc = 'upper left')
plt.show()

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')

img2 = cdf[img]

hist,bins = np.histogram(img2.flatten(),256,[0,256])
cdf = hist.cumsum()
cdf_normalized = cdf * float(hist.max()) / cdf.max()
plt.plot(cdf_normalized, color = 'b')
plt.hist(img2.flatten(),256,[0,256], color = 'r')
plt.xlim([0,256])
plt.legend(('cdf','histogram'), loc = 'upper left')
plt.show()
```


File Edit View Navigate Code Refactor Run Tools VCS Window Help pcv7 - inlab1.py
pcv7 inlab1.py
Project
main.py inlab.py inlab1.py
pcv7 C:\Users\Sanjana\PycharmP
venv library root
inlab.py
inlab1.py
main.py
pcv1.jpeg
pcv2.png
pcv3.png
External Libraries
Scratches and Consoles
Run: inlab1
Figure 1
5000
4000
3000
2000
1000
0
0
50
100
150
200
250
x=187.2 y=3941.
Version Control Run TODO Problems Terminal Python Packages Python Console Event Log
4:32 CRLF UTF-8 4 spaces Python 3.9 (pcv7)

```
C:\Users\Sanjana\Pycha                                    lab1.py
```

Packages installed successfully: Installed packages: 'matplotlib' (11 minutes ago)

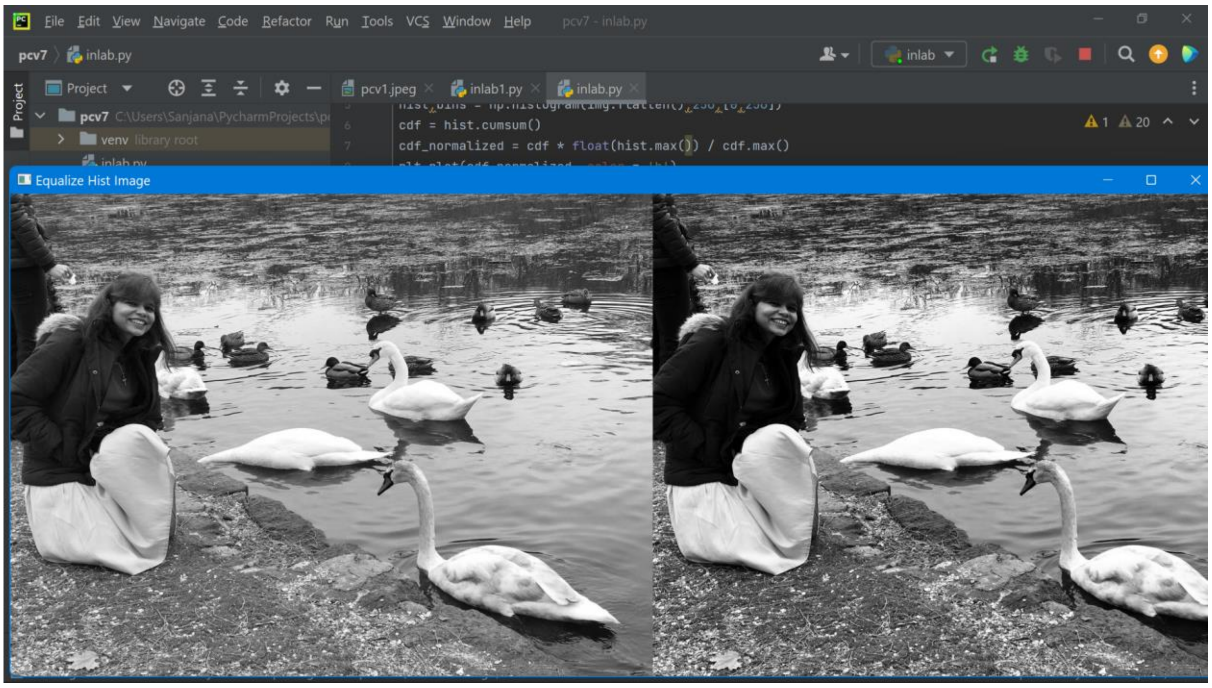


```
import numpy as np
```

```
C:\Users\Sanjana\PycharmProjec
```

```
Packages installed successfully: Installed packages: 'matplotlib' (18 minutes ago)
```

```
import numpy as np
import cv2 as cv
from matplotlib import pyplot as plt
img = cv.imread('pcv1.jpeg',0)
hist,bins = np.histogram(img.flatten(),256,[0,256])
cdf = hist.cumsum()
cdf_normalized = cdf * float(hist.max()) / cdf.max()
plt.plot(cdf_normalized, color = 'b')
plt.hist(img.flatten(),256,[0,256], color = 'r')
plt.xlim([0,256])
plt.legend(('cdf','histogram'), loc = 'upper left')

equ = cv.equalizeHist(img)
res = np.hstack((img,equ))
cv.imwrite('equ.png',equ)
cv.imshow('Equalize Hist Image',res)
cv.waitKey(0)
cv.destroyAllWindows()
```

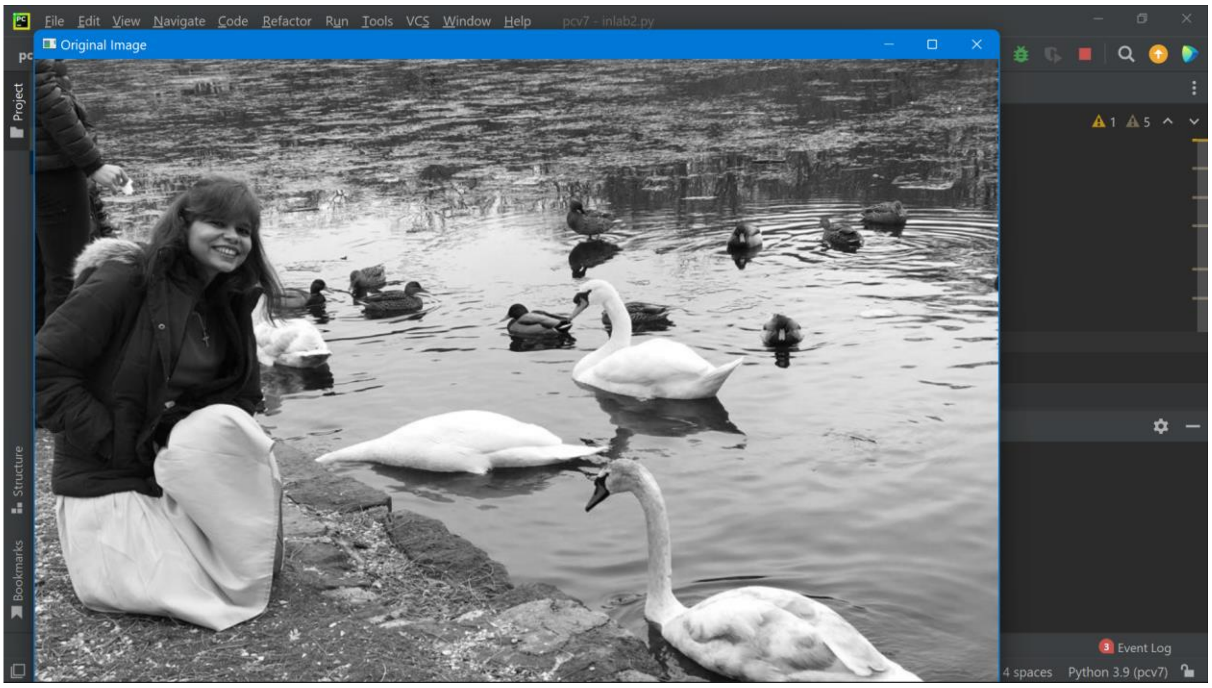

File Edit View Navigate Code Refactor Run Tools VCS Window Help pcv7 - inlab.py
pcv7 inlab.py
Project
pcv7 C:\Users\Sanjana\PycharmProjects\p
venv library root
pcv1.jpeg
inlab1.py
inlab.py
Equalize Hist Image

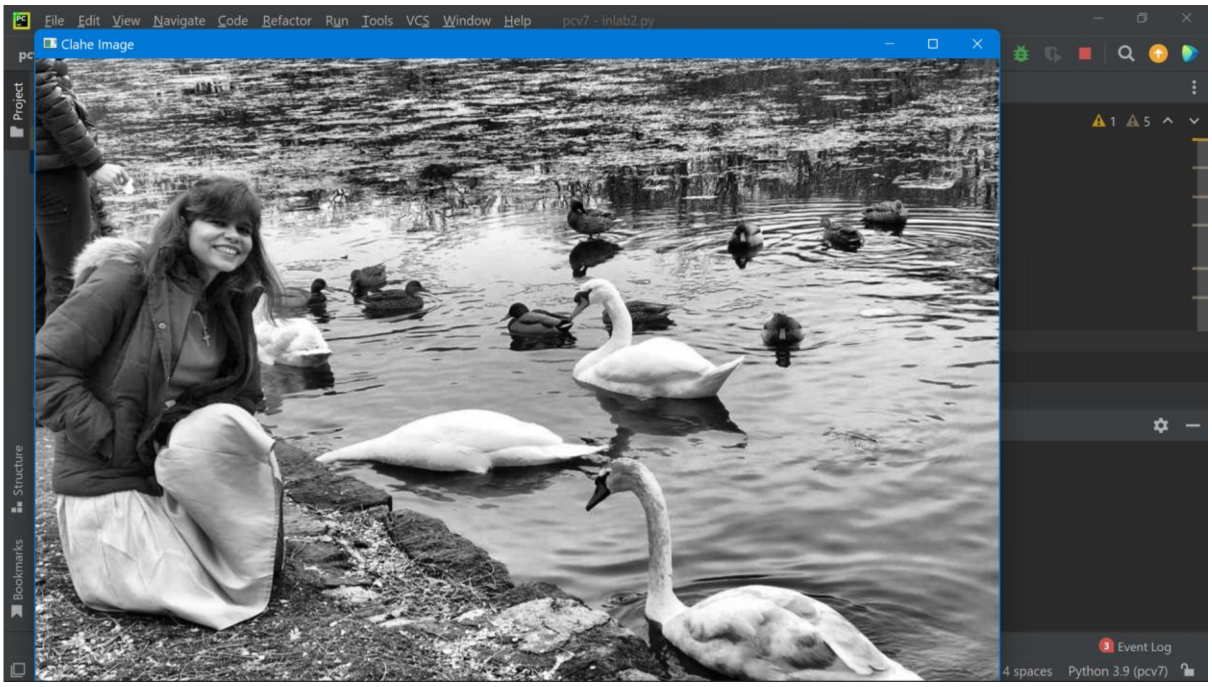
```
cdf = hist.cumsum()
cdf_normalized = cdf * float(hist.max()) / cdf.max()
```

2. Solve the problem of over-brightness using the adaptive histogram equalization by using the OpenCV CLAHE method

A.

```
import numpy as np
import cv2 as cv
img = cv.imread('pcv1.jpeg',0)

clahe = cv.createCLAHE(clipLimit=2.0, tileGridSize=(8,8))
cl1 = clahe.apply(img)
cv.imwrite('clahe.jpg',cl1)

cv.imshow('Original Image', img)
cv.imshow('Clahe Image',cl1)
cv.waitKey(0)
cv.destroyAllWindows()
```


File Edit View Navigate Code Refactor Run Tools VCS Window Help pcv7 - inlab2.py
Original Image
Project
Structure
Bookmarks
Event Log
4 spaces Python 3.9 (pcv7)


File Edit View Navigate Code Refactor Run Tools VCS Window Help pcv7 - inlab2.py
Clahe Image
Project
Structure
Bookmarks
Event Log
4 spaces Python 3.9 (pcv7)

## POSTLAB

1. Discuss at least two differences between the equalizeHist method and the CLAHE method of histogram equalization.

A. Histogram equalization is an image processing technique for adjusting the image's intensity. This enhances the contrast in an image.

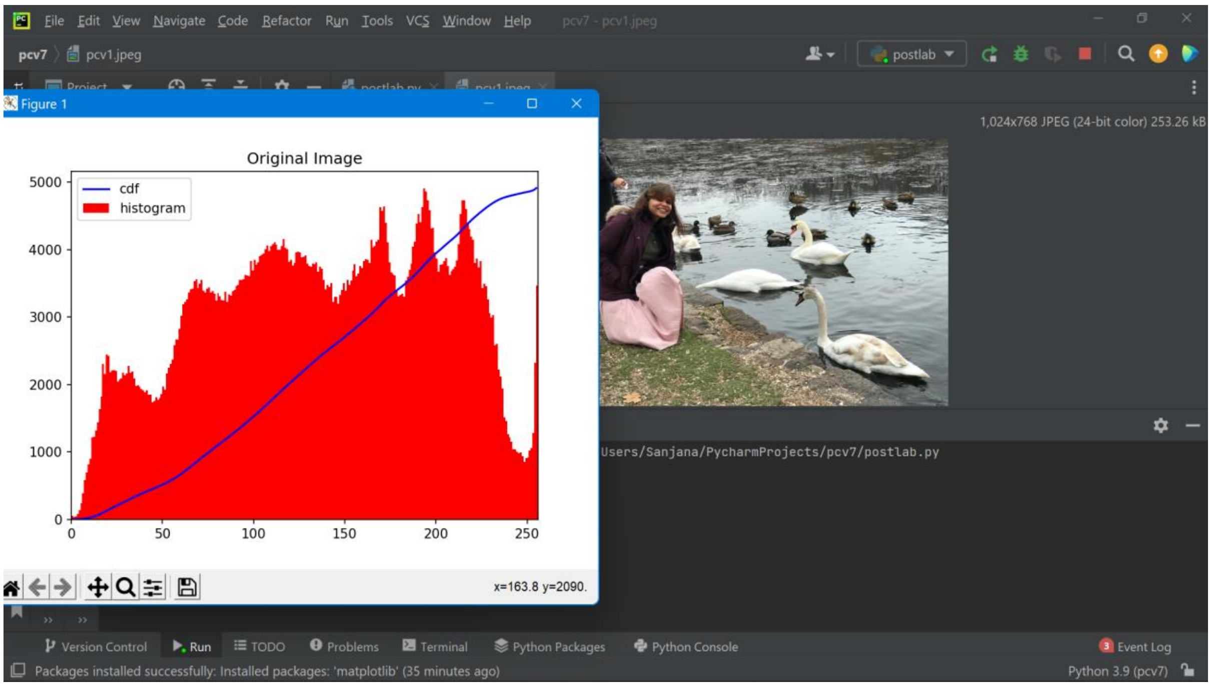
Behind the scenes, it can be explained by using a histogram. An equalized histogram means that the image uses all grey levels in equal proportions. Then, intensities are better distributed on the histogram.

Contrast limited AHE limits the contrast amplification to reduce amplified noise. It does so by distributing that part of the histogram that exceeds the clip limit equally across all histograms

2. Plot the histogram of the original image, the histogram of the histogram-equalized image using the equalizehist function, and the histogram of the image after applying the CLAHE method.

A.

```
import numpy as np
import cv2 as cv
from matplotlib import pyplot as plt
img = cv.imread('pcv1.jpeg',0)
hist,bins = np.histogram(img.flatten(),256,[0,256])
cdf = hist.cumsum()
cdf_normalized = cdf * float(hist.max()) / cdf.max()
plt.plot(cdf_normalized, color = 'b')
plt.hist(img.flatten(),256,[0,256], color = 'r')
plt.xlim([0,256])
plt.legend(('cdf','histogram'), loc = 'upper left')
plt.title('Original Image')
plt.show()
```



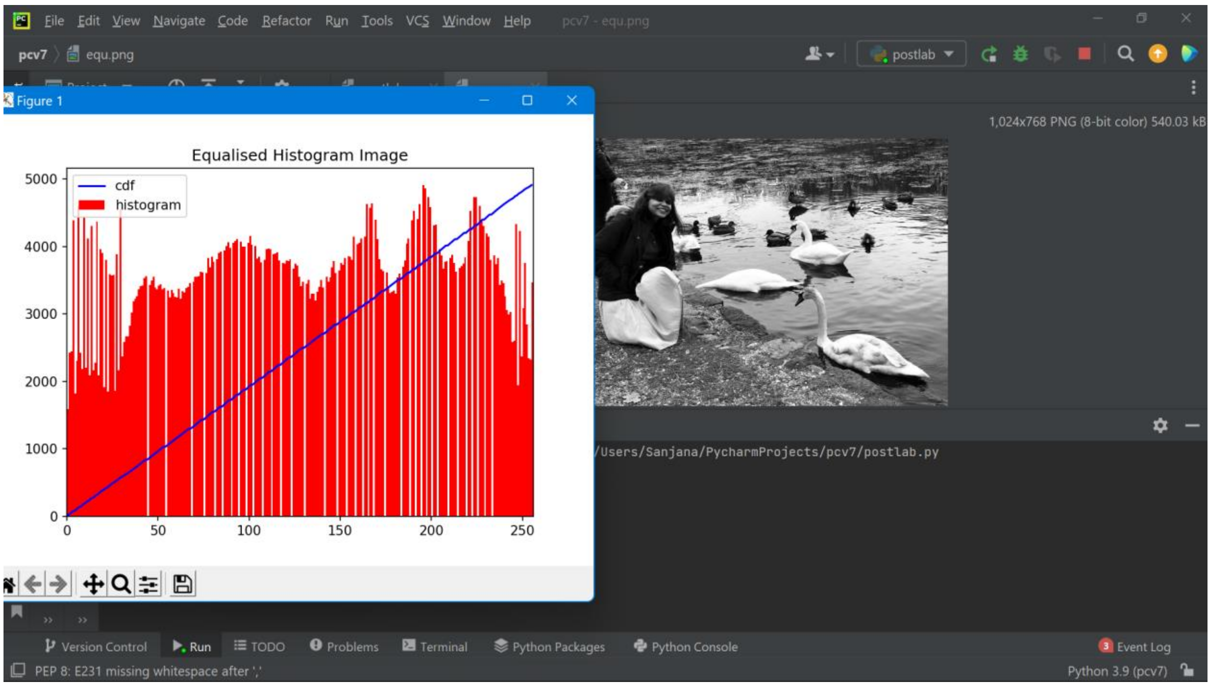
1,024x768 JPEG (24-bit color) 253.26 kB

```
Users/Sanjana/PycharmProjects/pcv7/postlab.py
```

Packages installed successfully: Installed packages: 'matplotlib' (35 minutes ago)

```
import numpy as np
import cv2 as cv
from matplotlib import pyplot as plt
```

```python
img = cv.imread('equ.png',0)
hist,bins = np.histogram(img.flatten(),256,[0,256])
cdf = hist.cumsum()
cdf_normalized = cdf * float(hist.max()) / cdf.max()
plt.plot(cdf_normalized, color = 'b')
plt.hist(img.flatten(),256,[0,256], color = 'r')
plt.xlim([0,256])
plt.legend(('cdf','histogram'), loc = 'upper left')
plt.title('Equalised Histogram Image')
plt.show()
```

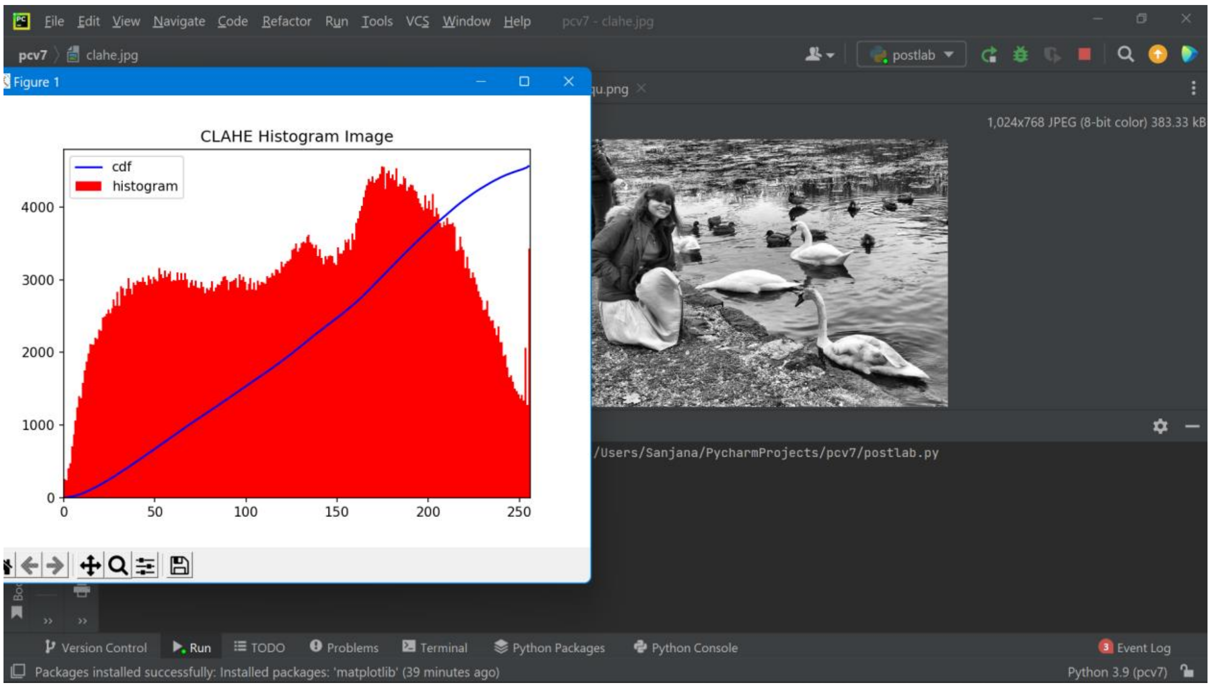


1,024x768 PNG (8-bit color) 540.03 kB

```
/Users/Sanjana/PycharmProjects/pcv7/postlab.py
```

PEP 8: E231 missing whitespace after ','

```python
import numpy as np
import cv2 as cv
from matplotlib import pyplot as plt
img = cv.imread('clahe.jpg',0)
hist,bins = np.histogram(img.flatten(),256,[0,256])
cdf = hist.cumsum()
cdf_normalized = cdf * float(hist.max()) / cdf.max()
plt.plot(cdf_normalized, color = 'b')
plt.hist(img.flatten(),256,[0,256], color = 'r')
plt.xlim([0,256])
plt.legend(('cdf','histogram'), loc = 'upper left')
plt.title('CLAHE Histogram Image')
plt.show()
```



```
/Users/Sanjana/PycharmProjects/pcv7/postlab.py
```

```
Packages installed successfully: Installed packages: 'matplotlib' (39 minutes ago)
```

3. Analyse the three histograms in question 2 of post-lab and discuss the differences between them.

A. Comparison shows the result of the results of contrast enhancement. From the three images it can be seen that the CLAHE method displays images more clearly than the other two methods, however it produces noise on image. To reduce the noise, we use median filter as filtering method. This method serves to remove noise generated by CLAHE method. The median filter works by calculating the value of each pixel, the pixel value at the center of the kernel coordinate with the median value of all pixels in the kernel.

CLAHE limits the range of the output image so that you don't get big outliers. In essence, it clips values if they get too huge. This is the usual method people use. There is no real harm but good benefits so there's no reason not to do it. I don't ever see straight AHE being used. CLAHE is performed by adapthisteq() in the Image Processing Toolbox.

There is also a histogram equalization function histeq() but it's worthless because it uses the histogram of the whole image. Anyone who's ever used histogram equalization can tell you that global histeq is no good. Histogram equalization is only good if you do it locally like CLAHE does.