IMAGE PROCESSING

FIRST ASSESMENT

IMPORT

imports

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

path = r'img.png'
img = cv2.imread("img.png")
```

READ A COLOR IMAGE

This function reads a dark color image

RGB

```
def rgbimage():
    cv2.imshow('RGB', img)
    cv2.waitKey(0)
    cv2.destroyAllWindows()
#call
 rgbimage()
```

OUT PUT

CONVERT INTO GRAY-SCALE

In this part, there is a function that converts the image to grayscale without storing it, and the first line converts it and stores it

gray

```
#new gray image with store
img2= cv2.imread(path, 0)
#function to convert image with no store
def grayimage():
    img = cv2.imread(path, 0)
    cv2.imshow('gray', img)
    cv2.waitKey(0)
    cv2.destroyAllWindows()
    plt.show()
#call
 grayimage()
```

OUT PUT

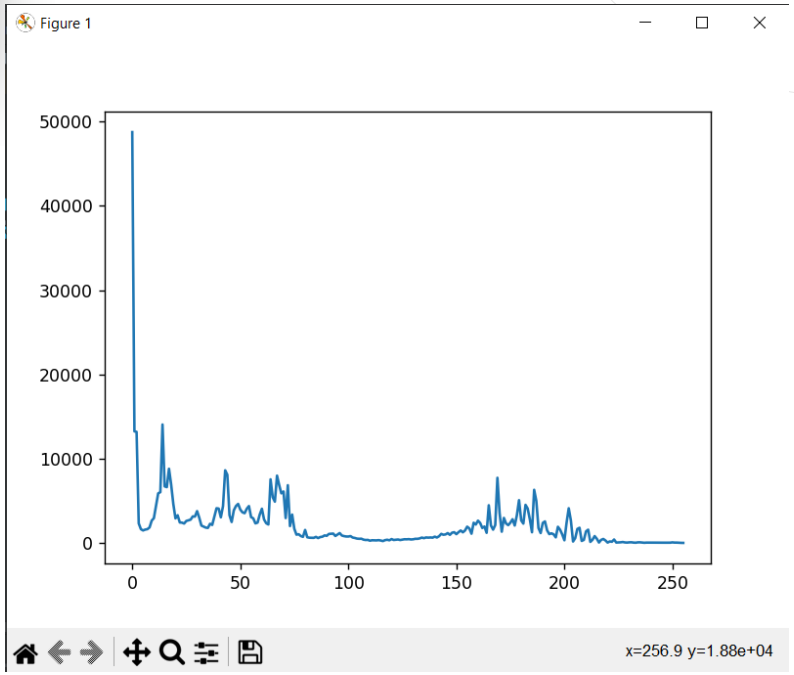
SHOW THE HISTOGRAM

This is the histogram of a color image, but it will not differ from a grayscale image

hestofor rgb

```python
def Hestogarmergb():
    img = cv2.imread('img.png',0)
    histr = cv2.calcHist([img],[0],None,[256],[0,256])
    plt.plot(histr)
    plt.show()
#call
Hestogarmergb()
```

OUT PUT



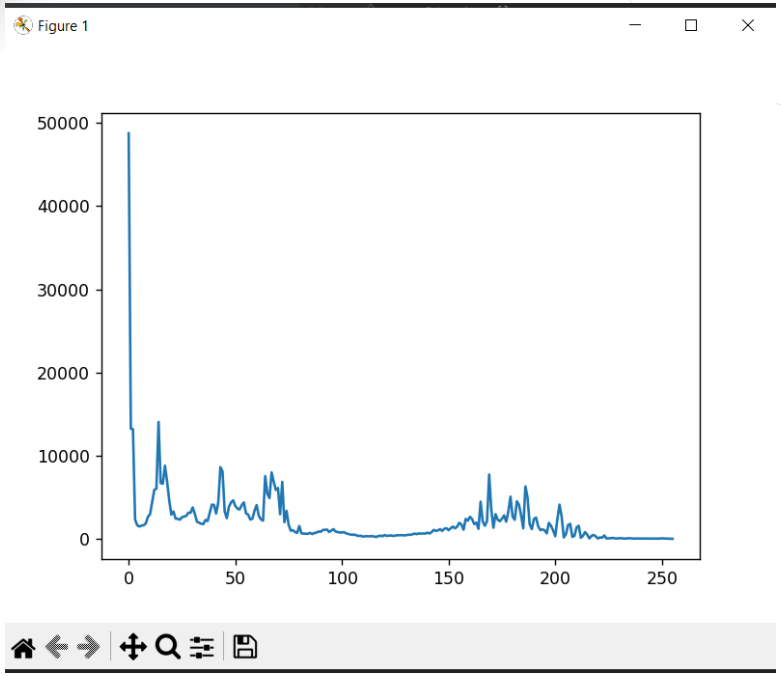
SHOW THE HISTOGRAM

This is the histogram of the grayscale image and it is similar to the color image

hestofor gray

```
def Hestogarmergbgray():
    histr = cv2.calcHist([img2],[0],None,[256],[0,256])
    plt.plot(histr)
    plt.show()
#call
Hestogarmergbgray()
```

OUT PUT



ENHANCE MANUALLY

Gamma law for color image correction

gama rgb

```python
#Enhnsmant function
def gammaCorrection(src, gamma):
    invGamma = gamma
    table = [((i / 255) ** invGamma) * 255 for i in range(256)]
    table = np.array(table, np.uint8)
    return cv2.LUT(src, table)

  #output show
def gamaimgprintrgb():
    gammaImg = gammaCorrection(img, 0.5)
    cv2.imshow('Gamma corrected image rgb', gammaImg)
    cv2.waitKey(0)
    cv2.destroyAllWindows()
#call
 gamaimgprintrgb()
```

OUT PUT

ENHANCE MANUALLY

Force law for grayscale correction

gama gray

```python
#Enhnsmant function
def gammaCorrection(src, gamma):
    invGamma = gamma
    table = [((i / 255) ** invGamma) * 255 for i in range(256)]
    table = np.array(table, np.uint8)
    return cv2.LUT(src, table)

  #output show
def gamaimgprintgray():
    gammaImg = gammaCorrection(img2, 0.5)
    cv2.imshow('Gamma corrected image gray', gammaImg)
    cv2.waitKey(0)
    cv2.destroyAllWindows()
#call
gamaimgprintgray()
```

OUT PUT

HISTOGRAM EQUALIZATION

This histogram is for a color image, which is not different from the gray as we will notice, and this function saves the output in a new image and stores it on the device turns it into gray

histogram equalization rgb

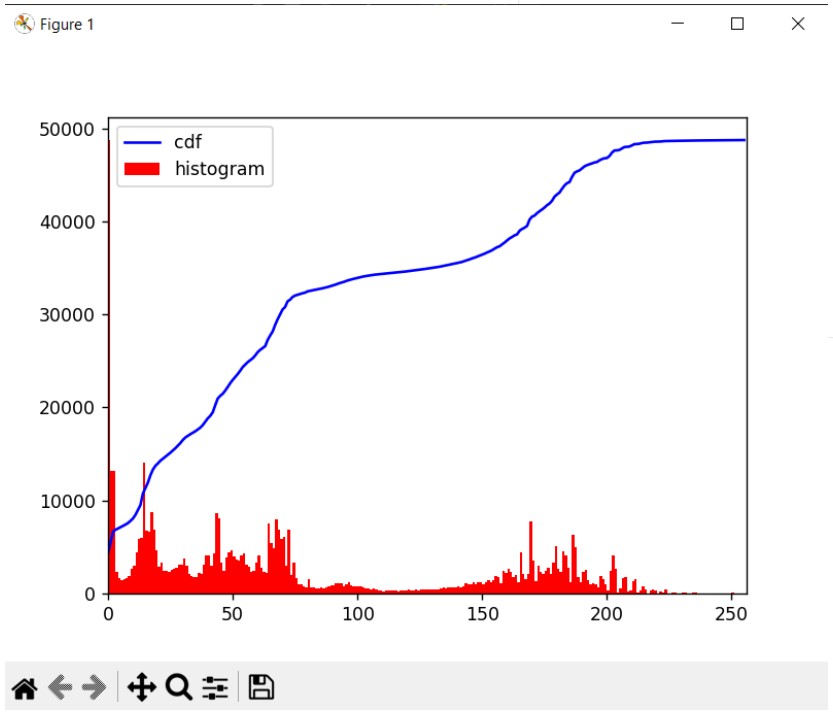
```python
def equlrgb():
    img = cv2.imread('img.png', 0)

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

    cdf = hist.cumsum()
    cdf_normalized = cdf * 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()
    img = cv2.imread('img.png', 0)
    equ = cv2.equalizeHist(img)
    res = np.hstack((img, equ))
    cv2.imwrite('resrgb.png', res)
#call
equlrgb()
```

OUTPUT



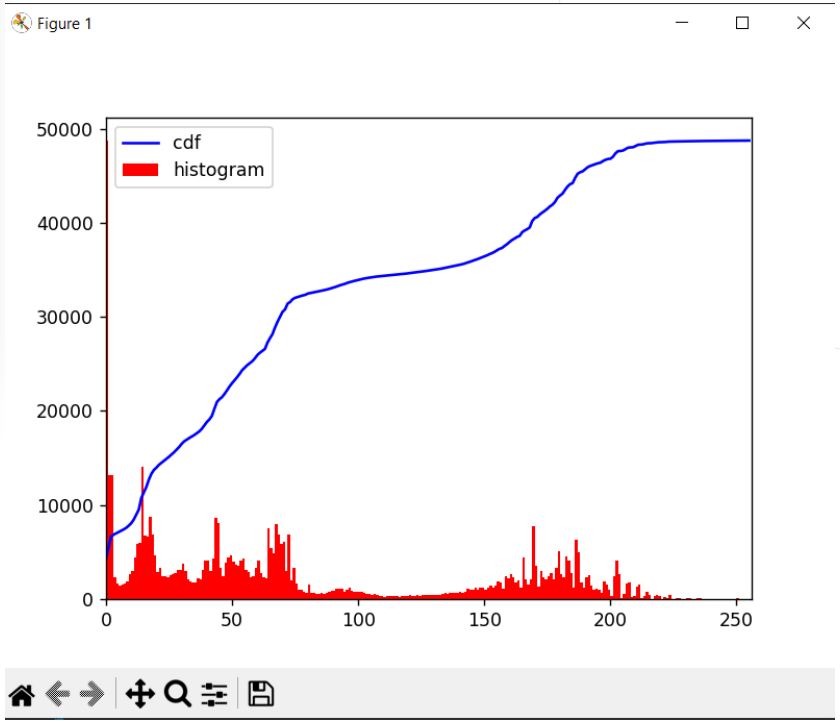
HISTOGRAM EQUALIZATION

This histogram is for a gray image and is no different from a colored one, and this pairing saves the output in a new image and stores it on the device

histogram equalization gray

```python
def equlgray():
    hist, bins = np.histogram(img2.flatten(), 256, [0, 256])
    cdf = hist.cumsum()
    cdf_normalized = cdf * 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()
    img = cv2.imread('img.png', 0)
    equ = cv2.equalizeHist(img)
    res = np.hstack((img, equ))
    cv2.imwrite('resgray.png', res)
#call
equlgray()
```

OUT PUT



HISTOGRAM EQUALIZATION

This pairing to display the image after equalization

histogram equalization rgb image

```
def equlrgbimage():
    img = cv2.imread("img.png",0)
    equ = cv2.equalizeHist(img)
    res = np.hstack((img, equ))
    cv2.imshow('image', res)
    cv2.waitKey(0)
    cv2.destroyAllWindows()
#call
equlrgbimage()
```

OUT PUT

HISTOGRAM EQUALIZATION

This pairing to display the image after equalization the gray image

histogram equalization gray image

```
def equlgrayimage():
    equ = cv2.equalizeHist(img2)
    res = np.hstack((img2, equ))
    cv2.imshow('image', res)
    cv2.waitKey(0)
    cv2.destroyAllWindows()
#call
equlgrayimage()
```

OUT PUT

NECESSARY FEATURES/VALUES

Gray image

necessary features/values gray

```
def histoInfogray():
    im = img2.ravel()
    m = statistics.mean(im)
    m1 = statistics.mode(im)
    m2 = statistics.median(im)
    v=statistics.variance(im,0)
    print("Gray image ")
    print("The Mean = " + str(m))
    print("The Mode = " + str(m1))
    print("The Median = " + str(m2))
    print("The Variance = " + str(v))
#call
histoInfogray()
```

OUT PUT

```
Gray image
The Mean = 76
The Mode = 0
The Median = 55.0
The Variance = 2763
```

MAIN

main

```
def main():
    rgbimage()
    grayimage()
    Hestogarmergb()
    Hestogarmergbgray()
    gamaimgprintrgb()
    gamaimgprintgray()
    equlrgb()
    equlgray()
    equlrgbimage()
    equlgrayimage()
    histoInfogray()

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