### Exercise 1: Image Representations and Point Operations

Url to Github: <https://github.com/caron1211/Computer-Vision-image-processing--exercises>

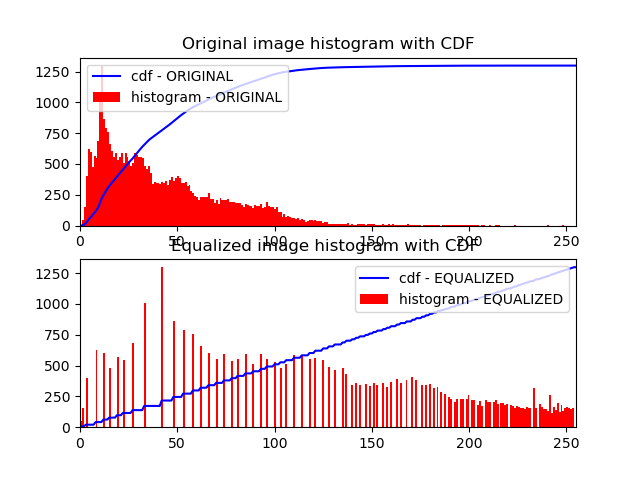
Histogram equalization for RGB :

I chose this image because it is dark image and can see that difference between the original image to the result

**Original:**



**The Histogram:**



**Result:**



Image Quantization for RGB (4 q) :

I choose this image have a lot of colors

Original:



Result:



### 4. Reading an image into a given representation

def imReadAndConvert(filename: str, representation: int) -> np.ndarray:  
 """  
 Reads an image, and returns and returns in converted as requested  
 :param filename: The path to the image  
 :param representation: GRAY\_SCALE or RGB  
 :return: The image object  
 The function used cv2.imread function to read the image.  
 It distinguished the case where the image is GRAYSCALE and RGB by representation param  
 In the case where the representation is RGB i convert from BGR to RGB  
 After converting to the matrix, I normalized all pixels to values between 0 and 1 by cv2.normalize  
  
 """

### 4.2 Displaying an image

def imDisplay(filename: str, representation: int):  
 """  
 Reads an image as RGB or GRAY\_SCALE and displays it  
 :param filename: The path to the image  
 :param representation: GRAY\_SCALE or RGB  
 :return: None  
 The function call to 'imReadAndConvert' and then use 'matplotlib.pyplot' library to show the image  
 """

### 4.3 Transforming an RGB image to YIQ color space

def transformRGB2YIQ(imgRGB: np.ndarray) -> np.ndarray:  
 """  
 Converts an RGB image to YIQ color space  
 :param imgRGB: An Image in RGB  
 :return: A YIQ in image color space  
 The function does dot product with the image and the matrix  
 """

### 4.4 Histogram equalization

def hsitogramEqualize(imgOrig: np.ndarray) -> (np.ndarray, np.ndarray, np.ndarray):  
 """  
 Equalizes the histogram of an image  
 :param imgOrig: Original Histogram  
 :return: imgEq:image after equalize , histOrg: the original histogram, histEq: the new histogram  
 The function get image performing a histogram calculation, I used the numpy function  
 and then computed the histogram cdf and then mapped the pixels in the image to the optimal cdf  
 """

### 4.5 Optimal image quantization

def quantizeImage(imOrig: np.ndarray, nQuant: int, nIter: int) -> (List[np.ndarray], List[float]):  
 """  
 Quantized an image in to \*\*nQuant\*\* colors  
 :param imOrig: The original image (RGB or Gray scale)  
 :param nQuant: Number of colors to quantize the image to  
 :param nIter: Number of optimization loops  
 :return: (List[qImage\_i],List[error\_i])  
 I divided the histogram into nQuant equal parts and then found the average in each section.  
 After finding the average, I changed all the pixels in that range to their average.  
 Repeat this process nIter times and at each stage find the mistake between the image we calculated and the original image  
  
 """

### 4.6 Gamma Correction

def gammaDisplay(img\_path: str, rep: int):  
 """  
 GUI for gamma correction  
 :param img\_path: Path to the image  
 :param rep: grayscale(1) or RGB(2)  
 :return: None  
 I used Trackbar of cv2 The values in the trackbar are between 0-100 and default is 50.  
 I converted this range to the [0-2] ( default is 1) and called tha value gamma.  
 Each time the user moved the range I updated the values of gamma powerful image  
 To exit GUI, click esc  
 """

