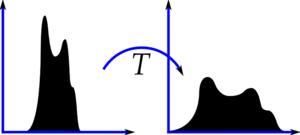
Introduction :-

When an object is place infront of bright light it front side becomes darker so there is imbalance in the image .We can't see our subject or point of focus clearly i.e there is difference in intensity . So to enhance the quality of image I have use Histogram Equalization which uses the concept of cumulative frequency and balances the intensity of pixels in an image.

**Histogram Equalization :-**

Consider an image whose pixel values are confined to some specific range of values only. For eg, brighter image will have all pixels confined to high values. But a good image will have pixels from all regions of the image. So you need to stretch this histogram to either ends (as given in below image, from wikipedia) and that is what Histogram Equalization does (in simple words). This normally improves the contrast of the image.

The method is useful in images with backgrounds and foregrounds that are both bright or both dark.

* Implementation of Histogram Equalization

**h(v) = round( ( ( cdf (v) - cdf min ) \* (L-1) ) / ( ( M \* N ) - cdf min) )**

**cdf min** =  minimum non-zero value of the cumulative distribution function

**L** = number of grey levels used(0-255)

**M** = width of image

**N** = height of image

**M \* N** = image's total number of pixels

**Code :-**

import cv2

import numpy as np

* In these part of code we have simply import our necessary packages require for our project.

original = cv2.imread('D:/TRF IP TASK/m4.jpg')

ori\_YUV=cv2.cvtColor(original,cv2.COLOR\_BGR2YUV)

* Here we import our image from the require path to where our input image resides on disk.
* Input image in the BGR form.
* Then in second line we convert our image from BGR to YUV scale.

ori\_YUV[:,:,0] = cv2.equalizeHist(ori\_YUV[:,:,0])

histori=cv2.cvtColor(ori\_YUV,cv2.COLOR\_YUV2BGR)

* Here we apply our Histogram Equalization algorithm to our converted image (i.e in YUV) .
* equalizeHist( ) is the function use to apply the Histogram Equalization method.
* Then after applying the algorithm we again convert the image back into BGR scale .

result=cv2.resize(histori,(700,500),interpolation = cv2.INTER\_AREA)

ori1=cv2.resize(original,(700,500),interpolation = cv2.INTER\_AREA)

* Here both lines of code represent the same meaning of resizing of image to fit the window size.

cv2.imshow('ORIGINAL',ori1)  
cv2.imshow('RESULT',result)

cv2.waitKey(0)

cv2.destroyAllWindows()

* Here we print the original and final image which was generated after applying Histogram Equalization .
* waitKey(0 ) is use to show the window until any key is press.
* destroyAllWindows() is use to destroy all the windows that are created.

Result :-

Original image :-



Final image :-

