ITMO University

Image Processing:
Lab1

Prepared by

Bassel Alshawareb

April 11, 2023

1. Introduction

In this lab, we are performing histogram and processing images using techniques used with histograms. We are considering in this lab: Histogram equalization, shift, extending. The last part of this lab is dedicated to calculating a profile and projection of the image.

2. Histograms

Histogram are a representation of the frequencies of the intensities in the image. It consists of 256 intensity values, for which is assigned the number of appearances of the intensity in the image.

Shifting:

Shifting means increasing or decreasing all intensity values in the image by a single number. The main objective for shifting is to increase or decrease the brightness of the image.

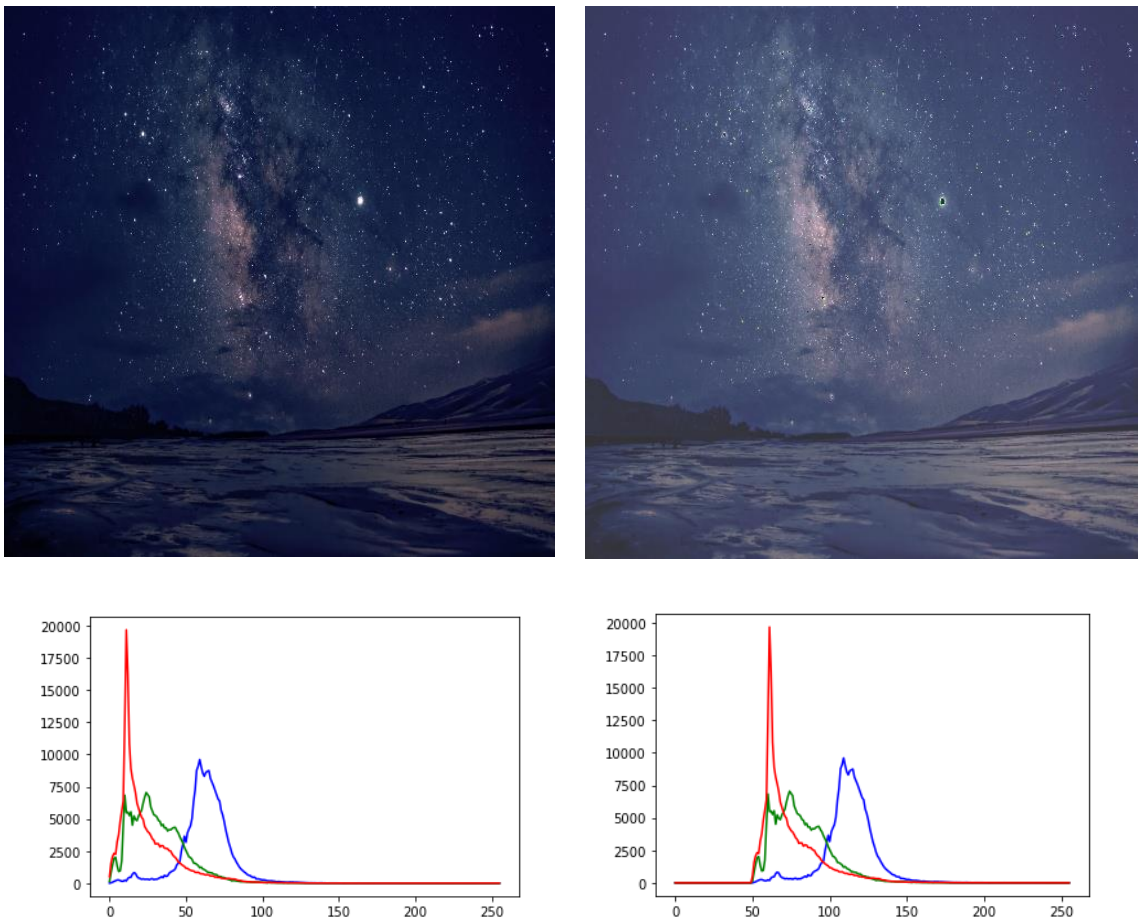

Figure 1 Applying histogram shift. a) The original dark image. b) A brighter version of the image after applying histogram shift. c) The histogram of the original image. d) The histogram of the resulting image.

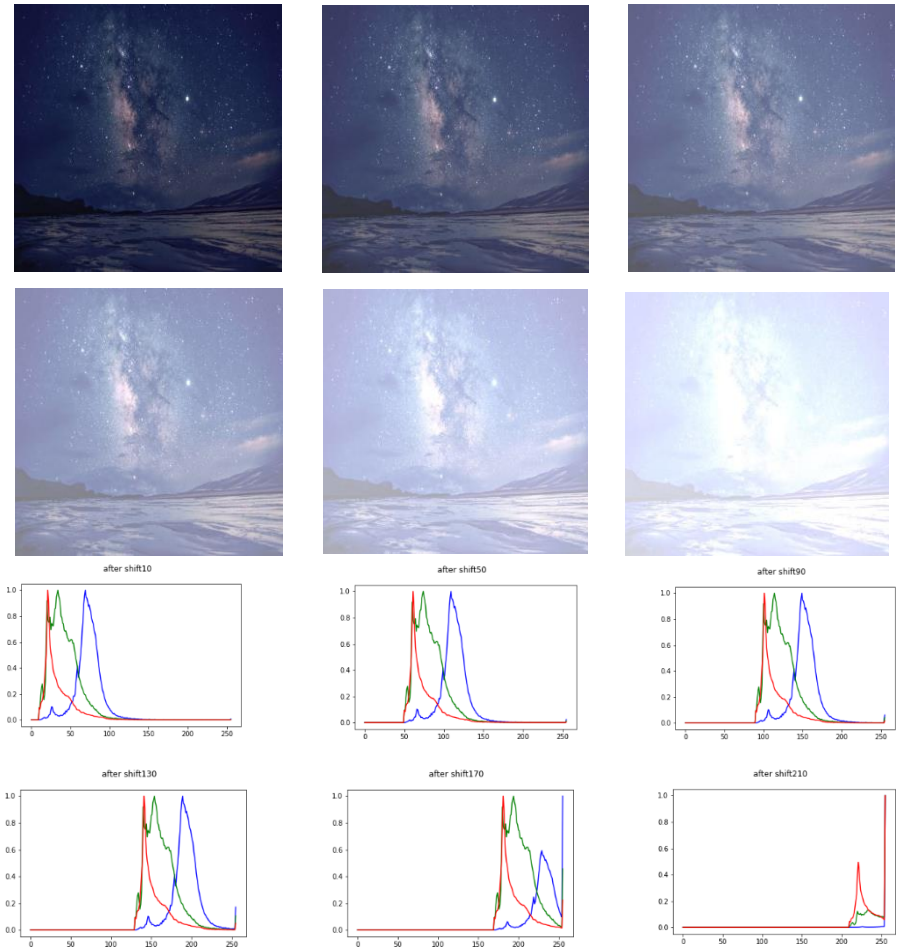

Figure 2 Applying multiple shift values on image. We can see that as the value gets bigger, there is a bigger chance of pushing the intensity values of the pixels outside of the range of 255. For this situation we need to handle overflow. To avoid the problem of overflow, we simply assign all values which are bigger than 255 to the value of 255. As the shift gets bigger, the frequency of the value 255 gets bigger.

Checking overflow:

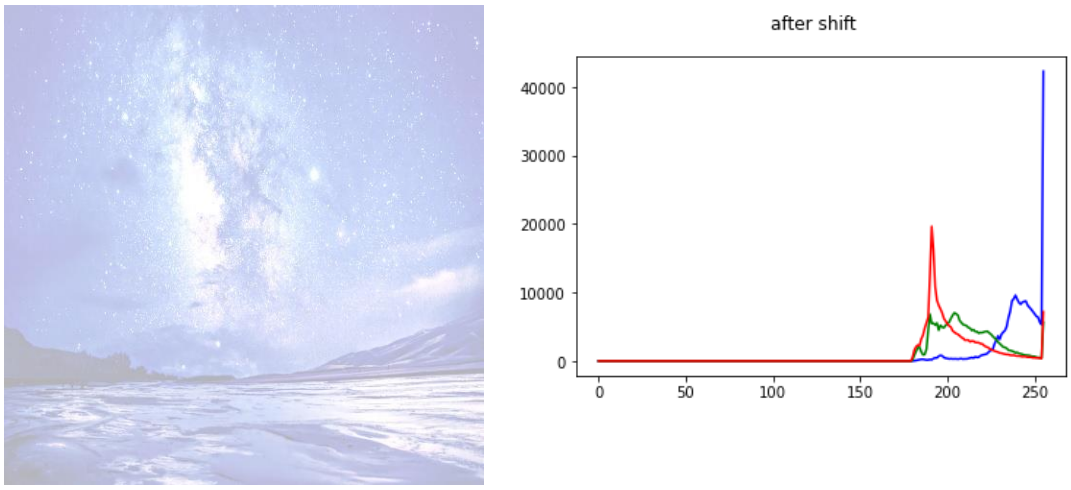

Figure 3 Applying high value shift to the image. A) The resulting image. B) The histogram of the resulting image.

We applied 180 value-shift to the image. We can see that the code takes in consideration the situation where the values get bigger than 255, and puts all the values which are bigger than it to 255.

Histogram Normalization:

Is an operation that maps the range of frequencies of the intensity to a probability distribution. In other words, it maps the values to a range between 0 and 1.

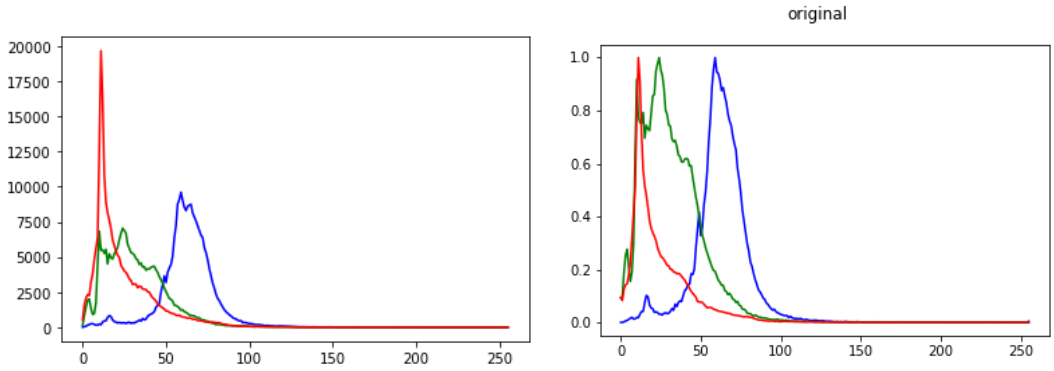

Figure 4 Histogram equalization. The image on the left represents the histogram of the original image. On the right is the equalized histogram.

Histogram Extending:

Here we have used two methods, one with which we have neglected the low frequencies in the image.
For the case, where we don't neglect low frequencies, We use this function to perform histogram extension:

$$I_{new} = \left(\frac{I - I_{min}}{I_{max} - I_{min}} \right)^{\alpha}$$

With values of Alpha smaller than 1, this function stretches low intensity values in the image along the total range of the image.

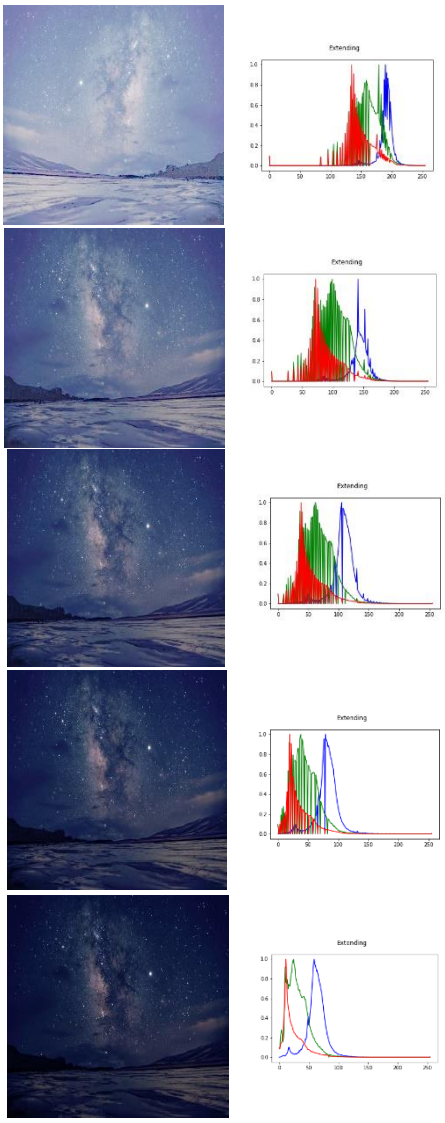

Figure 5 Performing histogram extension with multiple values of alpha. The images from up to down are the results of applying histogram extending starting from small alpha ending with alpha equals to 1.

We can realize that when Alpha equals to 1, the function outputs the same input image. While for small values of alpha, we can see distortion of the original distribution of image pixels.

Another method we used, doesn't take in consideration the small frequencies of intensities in the image. In this method, we calculating the minimum intensity values in the image, after neglecting intensity values with frequencies less than a threshold. We consider the same approach for calculating the highest intensity value. This we apply a simple mapping between the calculated bounding values and the range [0, 255]. Here we use alpha equals to 1.

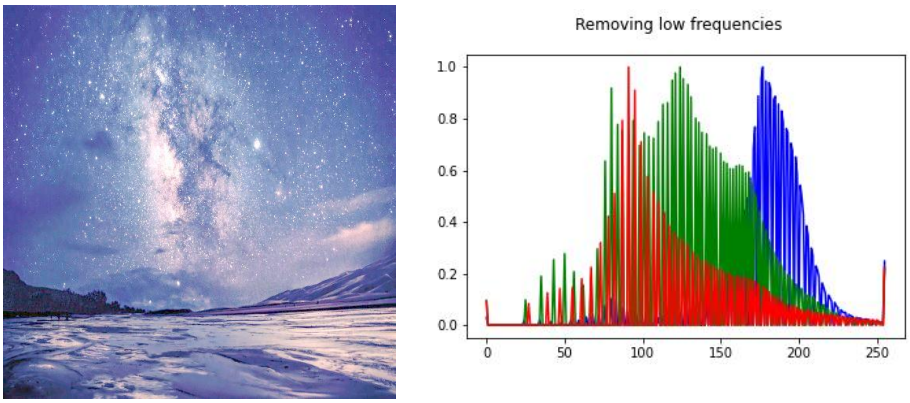

Figure 6 Histogram extending after removing low frequencies.

We can see that using this approach, we definitely result with more expanded distribution of the pixel along all the image. However, a disadvantage of this approach is that it results with high frequencies of 0, and 255 pixels, which don't belong to any distribution.

3. Profile

Profile is taking a line in the image and displaying its pixels.

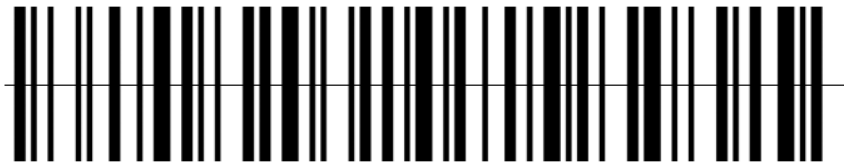

Figure 7 Bar code. The line indicates the place of which we want to take the profile.

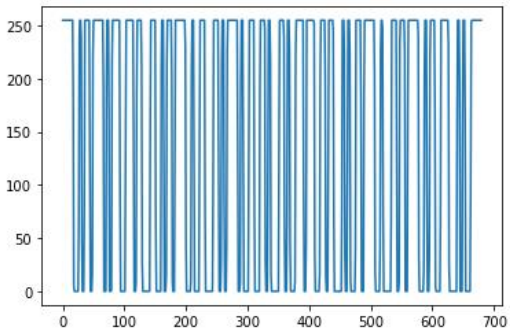

Figure 8 The profile of bar code image. The changes in intensities indicates the changes of color intensities along the profile.

4. Projection

Projection is simply summing the pixel values in either direction x or y.

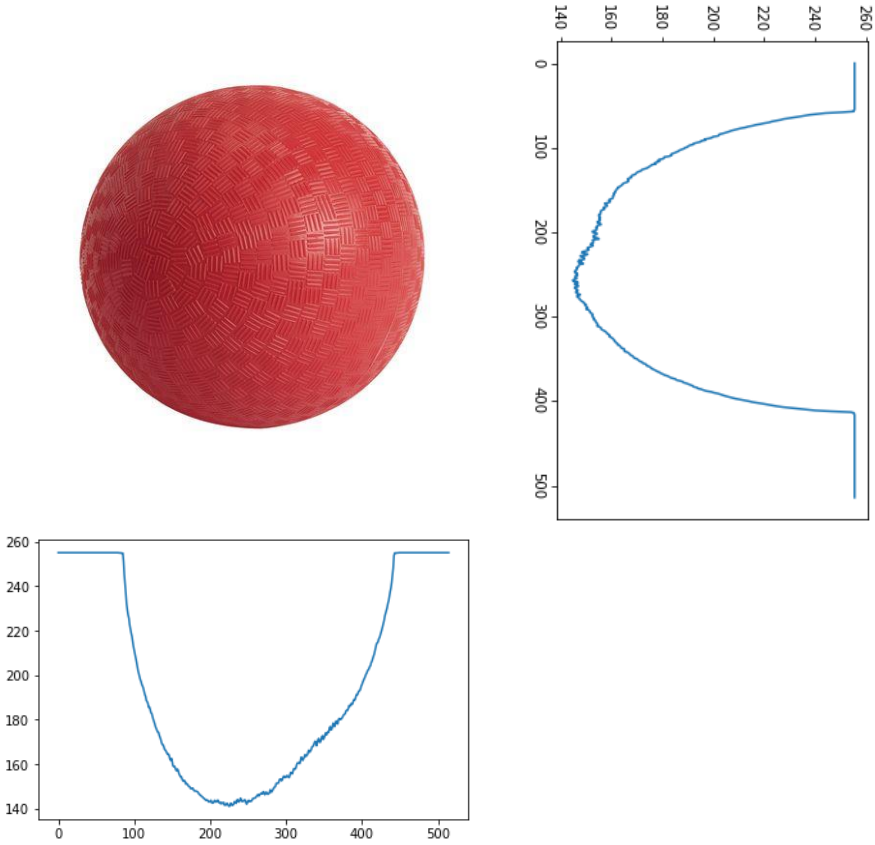

Figure 9 Image of ball with its projections along x and y axes

5. Questions

1. Image contrast refers to the difference in brightness or color between objects or regions in an image. It is a measure of how well the different parts of an image can be distinguished from each other. Contrast can be changed by adjusting the brightness and/or contrast settings of an image using photo editing software or by using techniques such as histogram equalization, which redistributes the brightness levels of an image to improve its contrast.

2. Image profiles and projections are useful tools for analyzing images and extracting information about their contents. An image profile is a plot of the pixel intensity values along a line or path through an image, which can reveal features such as edges, boundaries, and textures. Image projections are 1D histograms that summarize the distribution of pixel intensities in an image along a particular direction. They can be used to identify patterns, structures, or anomalies in an image, such as lines, circles, or clusters.

3. To find an object against the background, or more technically, segment an object in an image containing a uniform background, we can simply use threshold.

   Here we need to discuss multiple different cases:

   - If the background takes the biggest space in the picture, meaning that more than half of the pixels in the image belong to the background, We can find the threshold value by simply calculating the median value. The median value will be the value of the background pixels. Then we can perform the thresholding depending on that value.

   - If the object takes the biggest part of the image, then we can calculate the histogram and then apply one of the histogram segmentation techniques, for example, OTSU algorithm, where we fit gaussian distribution to the image histogram, and find the threshold value that splits the gaussian distributions of histogram values. Then we perform the thresholding.

6. Code:

```
# -*- coding: utf-8 -*-
"""
Created on Sat Mar 25 13:21:23 2023

@author: Bassel
"""

import cv2
import numpy as np
import matplotlib.pyplot as plt
import os

path = "C:/Users/Bassel/Documents/GitHub/Image-Processing/Histograms"
path_input = path + "/inputs"
path_output = path + "/outputs"

class myHist:
    id_plot=0
    def __init__(self, img=None,histSize=256, histRange=(0,256),
CONFIG="BGR", EQUALIZE = True):
        self.last_executed = ""
        self.histSize=256
        self.EQUALIZE = EQUALIZE
        if CONFIG=="BGR":
            a=[0,1,2]
        else:
            a=[2,1,0]
        self.order=a
        myHist.histSize=histSize
        myHist.histRange=histRange
        if img is not None:
            self.img=img.copy()
            self.calc(self.img)
        else:
            self.img=img

    def calc(self,img=None):
        print("Calculating Histogram")
        if img is None:
            if self.img is None:
                print("error")
                return
            img=self.img
        img_s=cv2.split(img)
        bHist=cv2.calcHist(img_s,[0],None, [256], (0, 256))
        gHist=cv2.calcHist(img_s,[self.order[1]],None, [self.histSize],
(0, 256))
        rHist=cv2.calcHist(img_s,[self.order[2]],None, [self.histSize],
(0, 256))
        self.img=img
        self.bH=bHist
        self.gH=gHist
        self.rH=rHist
        if self.EQUALIZE:
            self.equalize()
```

```
    def equalize(self):
        if self.last_executed == "":
            self.last_executed = "equalized"
        print("Equalizing")
        self.bH_not_normalized = self.bH
        self.gH_not_normalized = self.gH
        self.rH_not_normalized = self.rH
        max_b = np.max(self.bH)
        max_g = np.max(self.gH)
        max_r = np.max(self.rH)
        self.bH = self.bH/max_b
        self.gH = self.gH/max_g
        self.rH = self.rH/max_r
        self.EQUALIZE = True

    def shift(self, img=None, amount=50):
        if img is None:
            if self.img is None:
                print("error")
                return
            img = self.img
        I1=img.copy()

I1[:,:,0]=np.clip(I1[:,:,0].astype(np.int16)+amount,0,255).astype(np.ui
nt8)

I1[:,:,1]=np.clip(I1[:,:,1].astype(np.int16)+amount,0,255).astype(np.ui
nt8)

I1[:,:,2]=np.clip(I1[:,:,2].astype(np.int16)+amount,0,255).astype(np.ui
nt8)
        self.img = I1
        self.calc()

    def filter_high_frequencies(self, H):
        thresholded = np.where(H<= 10**(-2),10, H) #Filtering out small
frequencies
        i_min = 0
        print("length of H ", thresholded.shape[0])
        for i in range(thresholded.shape[0]):
            if thresholded[i] !=10:
                i_min = i
                break
        i_max = 255
        for i in range(thresholded.shape[0]):
            if thresholded[-1-i] !=10:
                i_max = 255-i
                break
        return float(i_min)/255, float(i_max)/255
    def extend(self, alpha = 0.5, REMOVE_LOW_FREQUENCY = True):
        if REMOVE_LOW_FREQUENCY:
            self.last_executed = "Extended"
        else:
            s_alpha = str(alpha)
            list_s_alpha = s_alpha.split('.')
            alpha_for_writing = '_'.join(list_s_alpha)
```

```
            self.last_executed = "Extended_with_Alpha" +
alpha_for_writing
        self.alpha = alpha
        I = self.img.astype(np.float64)/255
        Ib = I[:,:,0]
        Ig = I[:,:,1]
        Ir = I[:,:,2]
        Iout = []
        if self.EQUALIZE:
            if REMOVE_LOW_FREQUENCY:
                #removing low frequicies
                Ib_min, Ib_max = self.filter_high_frequencies(self.bH)
                Ig_min, Ig_max = self.filter_high_frequencies(self.gH)
                Ir_min, Ir_max = self.filter_high_frequencies(self.rH)
            else:
                Ib_min, Ib_max = np.min(Ib), np.max(Ib)
                Ig_min, Ig_max = np.min(Ig), np.max(Ig)
                Ir_min, Ir_max = np.min(Ir), np.max(Ir)

            #Extend b
            Ib_extended = ( np.clip((255*((Ib-Ib_min)/(Ib_max -
Ib_min))**alpha),0,255) ).astype(np.uint8)
            Iout.append(Ib_extended)
            #Extend g
            Ig_extended = ( np.clip((255*((Ig-Ig_min)/(Ig_max -
Ig_min))**alpha),0,255) ).astype(np.uint8)
            Iout.append(Ig_extended)
            #Extend r
            Ir_extended = ( np.clip((255*((Ir-Ir_min)/(Ir_max -
Ir_min))**alpha),0,255) ).astype(np.uint8)
            Iout.append(Ir_extended)
            self.img = cv2.merge(Iout)
            print("Extend done")
            self.calc(self.img)


    def show(self,name=None):
        I= cv2.cvtColor(self.img, cv2.COLOR_BGR2RGB)
        if name is None:
            name="number"+str(myHist.id_plot)
        image_name = name + "_" + self.last_executed
        image_path = path_output + "/" + "images"
        hist_path = path_output + "/" + "Histograms"
        try:
            os.mkdir(image_path)
        except IOError:
            pass
        try:
            os.mkdir(hist_path)
        except IOError:
            pass

        plt.figure(myHist.id_plot)
        t=range(256)
        plt.plot(t,self.bH, color="blue")
        plt.plot(t,self.gH, color="green")
        plt.plot(t,self.rH, color="red")
        plt.suptitle(name)
```

```
        plt.savefig(hist_path + "/" + image_name  + "_Histogram" +
".png")
        plt.show()
        myHist.id_plot=myHist.id_plot+1
        plt.figure(myHist.id_plot)
        plt.imshow(I)
        plt.show()
        plt.imsave(image_path + "/" + image_name + ".jpg", I)
        myHist.id_plot=myHist.id_plot+1

def profile(img, x):
    return img[x,:]

def project_(img,xy):
    return np.sum(img,xy)/(img.shape[(xy+1)%2])



if __name__ == "__main__":
    I=cv2.imread(path_input + '/dark_sky.jpg')
    img=I.copy()
    img=cv2.resize(img,(500,500))
    I=cv2.resize(I,(500,500))

    #Parameters to control histogram
    histSize = 256
    histRange = (0, 256)


    #histogram of the image
    H00=myHist(I)
    H00.equalize()
    H00.show("original")


    #Histogram of the shifted image
    shift_amounts=range(10,240,40)
    for shift_amount in shift_amounts:
        H00=myHist(I)
        H00.shift(amount= shift_amount)
        H00.show("after shift"+str(shift_amount))

    #extend with low frequencies
    alphas = range(10,1, -2)
    for alpha in alphas:
        alpha = alpha/10
        H2=myHist(I)
        H2.extend(alpha = alpha, REMOVE_LOW_FREQUENCY = False)
        H2.show("Extending")

    #extend with low frequencies filtering
    H2=myHist(I)
    H2.extend(REMOVE_LOW_FREQUENCY=True)
    H2.show("Removing low frequencies")

    #Extracting profile
    orr=cv2.imread(path_input + "/ballr.jpg", cv2.IMREAD_GRAYSCALE)
    img=orr.copy()
```

```
x_prof=round(img.shape[0]/2)
out=profile(img, x_prof)
cv2.line(orr,(10,x_prof), (img.shape[1], x_prof), color=[0])

plt.figure()
plt.plot(np.array(range(img.shape[1])), out)
cv2.imwrite(path_output + "/profile.png", orr)
plt.show()
cv2.imshow("profile", orr)
cv2.waitKey(0)
cv2.destroyAllWindows()


#projection
xy=1
proj=project_(img,xy)
t=np.asarray(range(img.shape[(xy+1)%2]))
plt.figure()
plt.plot(t,proj)
plt.show()
```