**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.

1. **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.

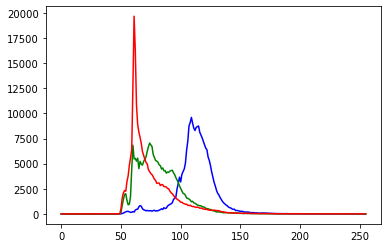
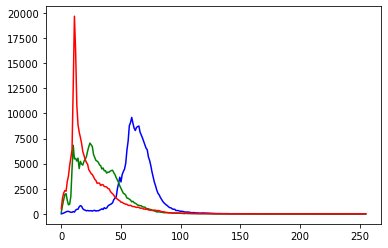


Figure Applying histogram shift. a) The original dark image. b) A brighter version of the image after applying histogram shift. c) The histogram of the original image. d) The histogram of the resulting image.

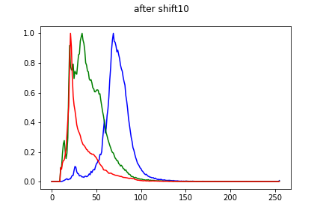
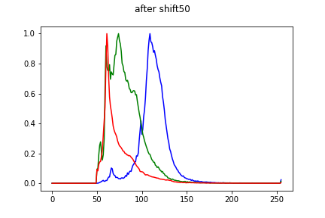
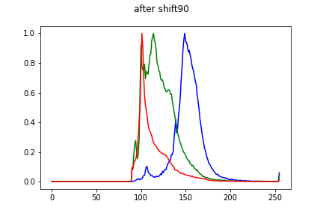
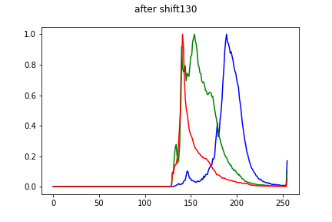
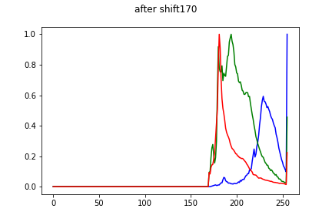
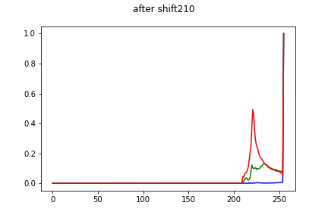
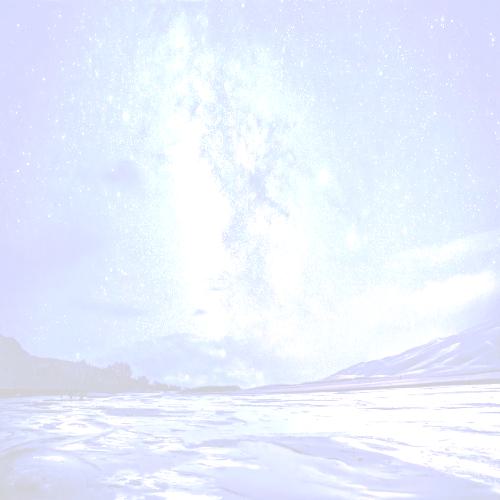


Figure Applying multiple shift values on image. We can see that as the value gets bigger, there is a bigger chance of pushing the intensity values of the pixels outside of the range of 255. For this situation we need to handle overflow. To avoid the problem of overflow, we simply assign all values which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

**Checking overflow:**

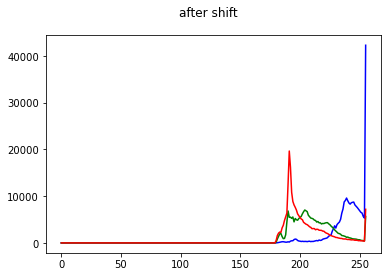
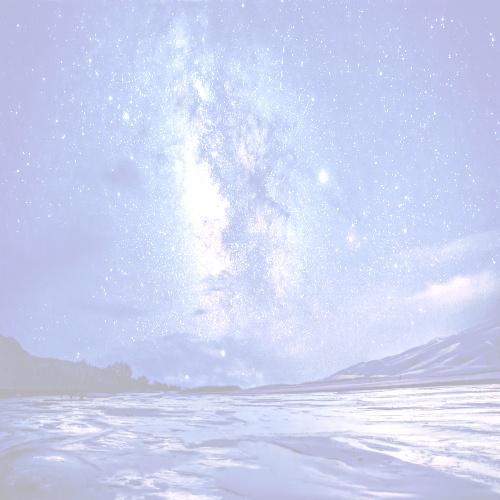


Figure Applying high value shift to the image. A) The resulting image. B) The histogram of the resulting image.

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.

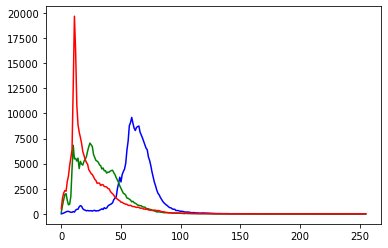
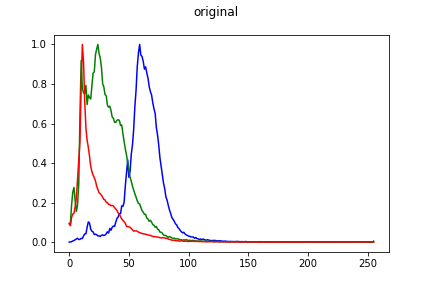


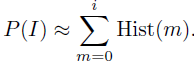
Figure Histogram equalization. The image on the left represents the histogram of the original image. On the right is the equalized histogram.

**Histogram Equalization:**

To perform equalization, we apply the following transformation:



Where P(I) is the accumulative histogram and it is derived from this formula:



This transformation could be referred to as uniform transformation, since it distributes the histogram uniformly over the range of minimum and maximum values. It can bee seen that the subtraction of Imin from Imax maps the probability function P(I) to the required range of intensity values.

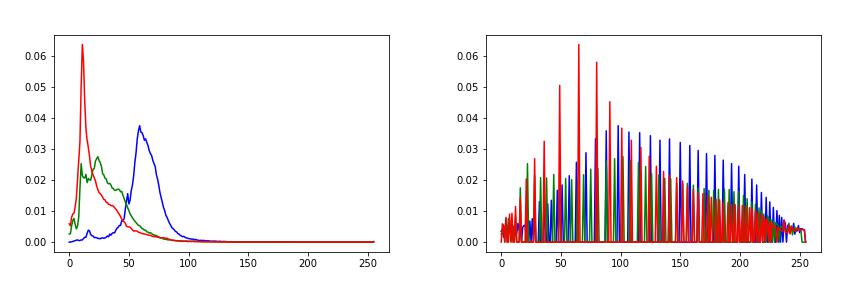


Figure Histogram equalization. a) The original Image. b) The equalized image. c) the histogram of the original image. d) The histogram after equalization.

By applying histogram equalization, we can make sure that intensity values will be uniformly distributed over the image. To see how this works, we can take a look at figure 6. The accumulated histogram maps the low intensity values over the whole range of values. This is because most intensities in the original histogram are low.

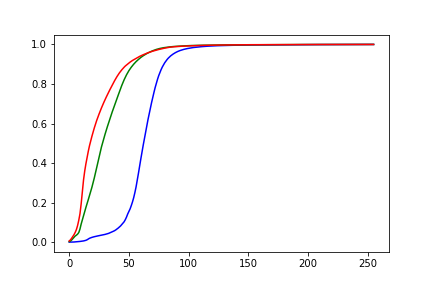


Figure The accumulated histogram.

**Exponential Transformation:**

The exponential transformation uses the following formula:



Where P(I) is the accumulative histogram.

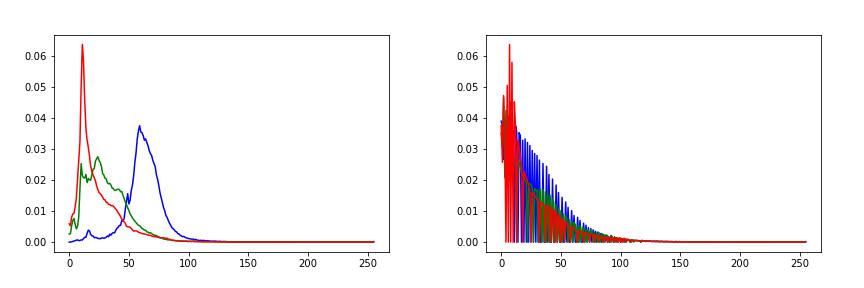


Figure Exponential transformation.

Alpha represents a value that maps the output of the logarithm to image intensity values. Since alpha is in the denominator, the more we decrease it, the bigger the output of the logarithm is. Therefore, as we are decreasing alpha values, the histogram is stretching more.

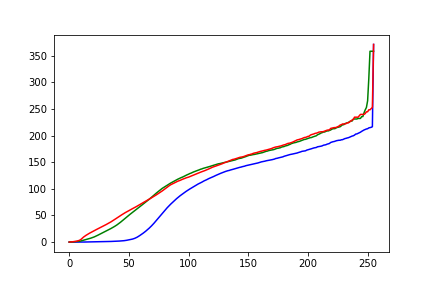
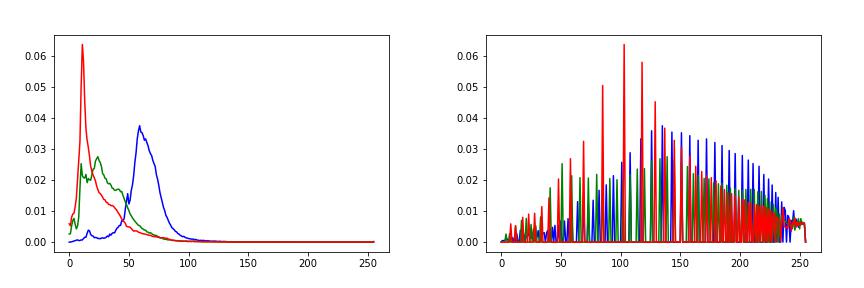
****

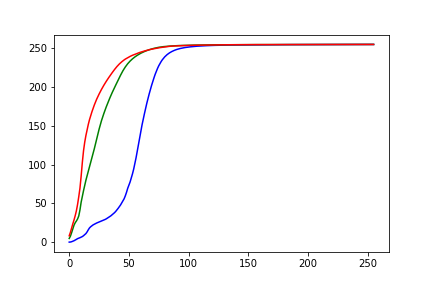
Figure The mapping function of the exponential transformation.

Since the histogram of the image is rich with low intensity values, we can deduce that it is not a good idea to shrink low intensity values into a smaller range of intensities. In our case, the exponential transformation function does exactly that, which leads to a darker image with less contrast.

**Transformation of 2/3 degree:**



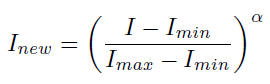
We can see that the distribution of pixel intensities follows exponential shape, and this results is very similar to the uniform transformation that we applied earlier.



**Histogram Extending:**

Here we have used two methods, one with which we have neglected the low frequencies in the image.

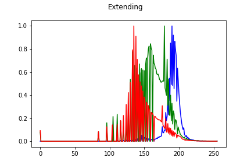
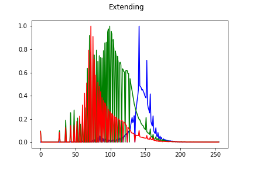
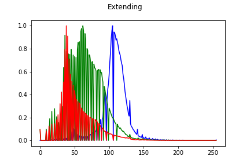
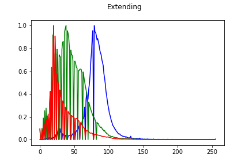
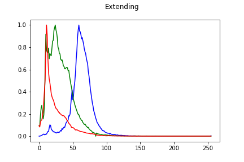
For the case, where we don’t neglect low frequencies, We use this function to perform histogram extension:



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

Figure 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.

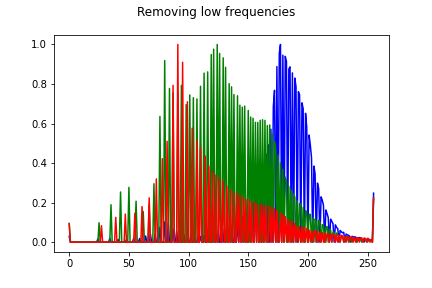


Figure Histogram extending after removing low frequencies.

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.

1. **Profile**

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

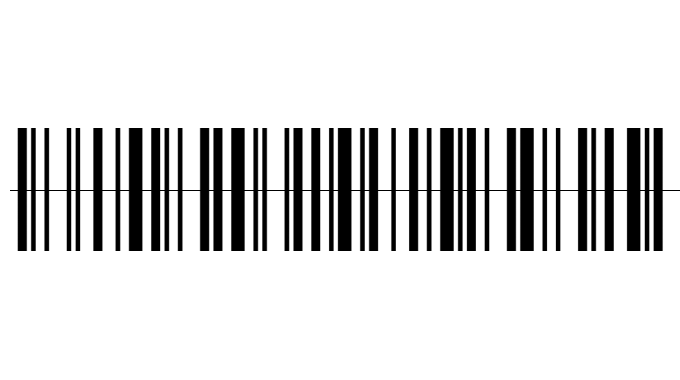


Figure Bar code. The line indicates the place of which we want to take the profile.

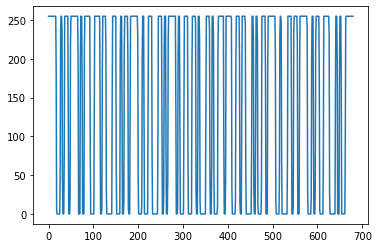


Figure The profile of bar code image. The changes in intensities indicates the changes of color intensities along the profile.

1. **Projection**

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

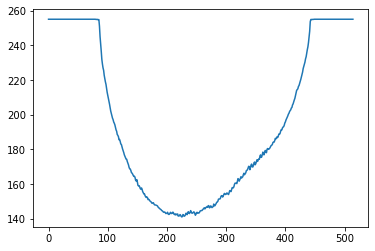
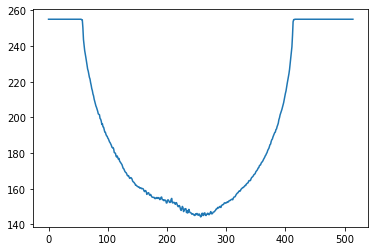


Figure Image of ball with its projections along x and y axes

1. **Questions**
2. 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.
3. 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.
4. 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.

1. **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.uint8)

I1[:,:,1]=np.clip(I1[:,:,1].astype(np.int16)+amount,0,255).astype(np.uint8)

I1[:,:,2]=np.clip(I1[:,:,2].astype(np.int16)+amount,0,255).astype(np.uint8)

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()