Zehra

1. General Properties

I used opencv, numpy and matplotlib libraries. Size of the test image is 150×300. Also I prefer an image which has low contrast.

2. Functions

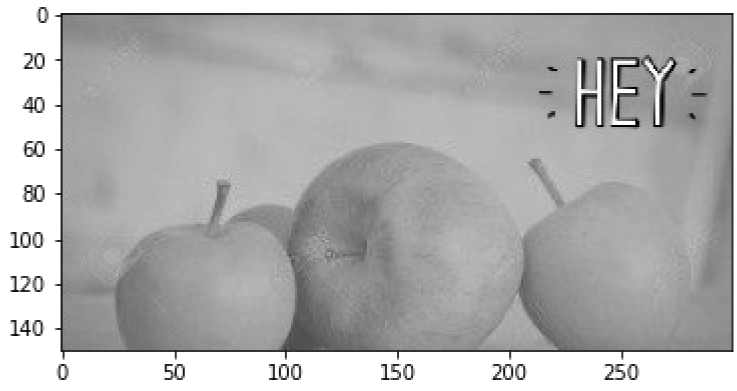
2.1. Read and Convert Gray Function

Parameter

- path: the location of the image file in the system

In this function we load an image in grayscale mode.

```
def read_and_convertgray(path):
   imgray=cv.imread(path,cv.IMREAD_GRAYSCALE)
   return imgray
```



2.2. Show Image Functions

Parameter

- img: image that we load using opencv

In this function we display an image in grayscale mode using matplotlib pyplot.

```
# SHOW IMAGE
def show_img(img):
   plt.imshow(img, cmap = 'gray')
   plt.show()
```

2.3. Image Size Function

Parameter

- img: image that we load using opencv

Return Value

- img.shape[0]: height of image
- img.shape[1]: width of image

In this function we get dimensions of the image using the shape function.

```
# GET SIZE OF IMAGE
def img_size(img):
   #height ,width
   return img.shape[0],img.shape[1]
```

2.4. Get, set Specific Pixel

2.4.1 Get Specific Pixel

Parameters

- img: image that we load using opencv
- row: height of image
- col: width of image

Return Value

- pixel: pixel value that we get by giving coordinates

We get the pixel where the column and row values are specified. We will use this function when flipping the image.

```
def get_specific_pixel(img,row,col):
   pixel= img[col,row]
   #print (pixel)
   return pixel
```

2.4.1 Set Specific Pixel

Parameters

- img: image that we load using opencv
- row: the row we want to change
- col: the column we want to change

Return Value

- img: image that a specific pixel changed

We set the pixel where the column and row values are specified. We will use this function when flipping the image.

```
def set_specific_pixel(img,row,col,pix):
   img[col,row]=pix
   return img
```

2.5. Flip Image Function

Parameters

- img: image that we load using opencv
- axis: type of flip. It is a string value that can be a “horizontal” or “vertical”.

Return Value

- img_new: New image that rotated horizontally or vertically

Our purpose is to flip a 2D array around vertically or horizontally. First of all, we subtract the height and width values. Because in python, numbers start from 0. We have two option for rotation.

2.5.1. Horizontal Flip

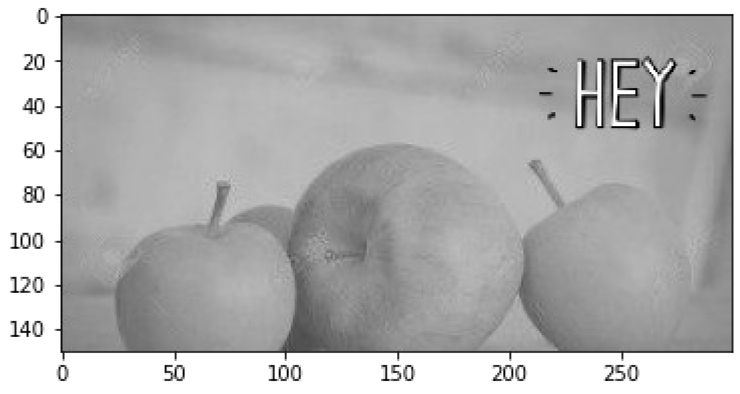
In horizontal flip, the first for loop visits every row value. The second for loop goes until half the width. We are getting the left pixel value with x and y. Then we find the right pixel that is horizontally symmetrical to the left pixel. If we subtract x from width we can find the exact position that we want. Then we change the pixel values in these two positions using set_specific_pixel function.

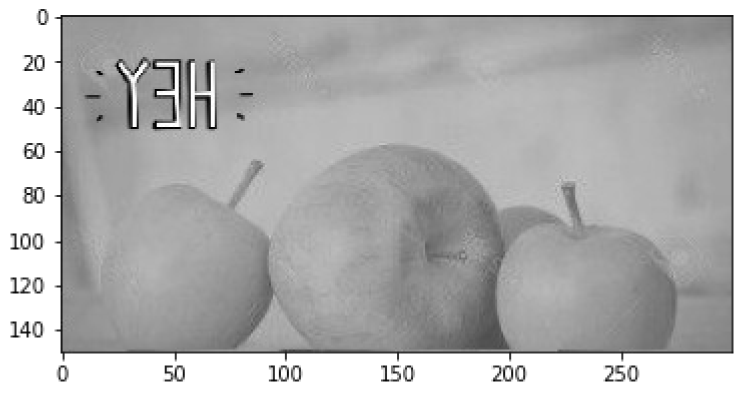
```
if axis == "horizontal":
       for y in range(height):
           for x in range(width//2):

               left = get_specific_pixel(img, x, y)
               right = get_specific_pixel(img, width-x, y)

               img_new0 = set_specific_pixel (img, width-x, y, left)
               img_new = set_specific_pixel (img_new0, x, y, right)

       return img_new
```





2.5.2. Vertical Flip

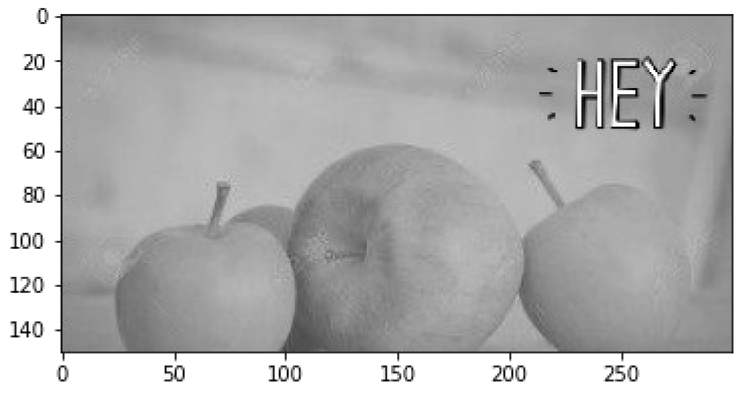
In vertical flip, the first for loop visits every column value. The second for loop goes until half the height. We are getting the bottom pixel value with x and y. Then we find the up pixel that is vertically symmetrical to the bottom pixel. If we subtract y from height we can find the exact position that we want. Then we change the pixel values in these two positions using set_specific_pixel function.

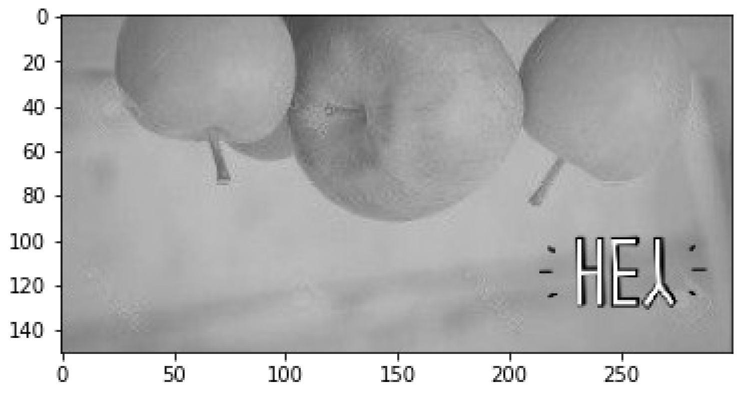
```
elif axis == "vertical":
       for x in range(width):
           for y in range(height//2):

               bottom = get_specific_pixel(img, x, y)
               up = get_specific_pixel(img, x, height-y)

               img_new0 = set_specific_pixel (img, x, height-y, bottom)
               img_new = set_specific_pixel (img_new0, x, y, up)

       return img_new
```





2.6. Generate Histogram Function

Parameters

- img: image that we load using opencv

Return Value

- hi: array of histogram values

A pixel can have an intensity value in range 0 to 255. So we need 256 values to show the histogram. We are creating an array with a size of 256. It contains zeros. We are visiting every pixel in the image and adding one because we are only counting the intensity values of each pixel(we have a grayscale image).

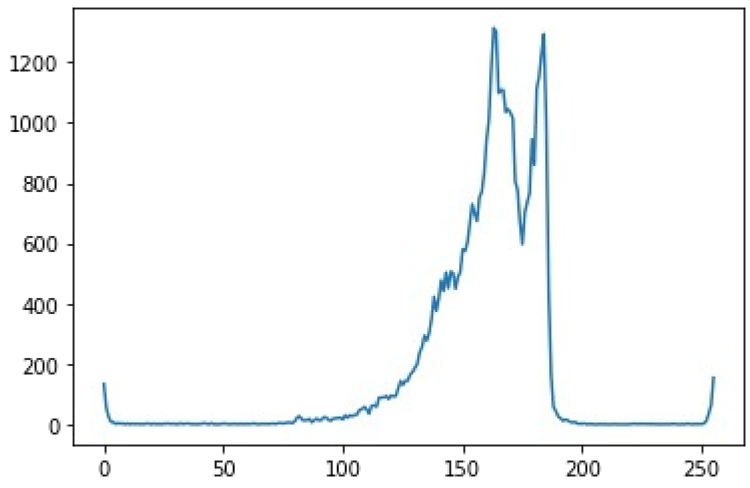
We have two different versions of the generate histogram function. First one is taking a 2 dimensional array ,second one taking a flatten image(1D).

```
def generateHistogram(img):
   hi= [0]*256
   rows= img.shape[0]
   cols= img.shape[1]
   for r in range(rows-1):
       for c in range(cols-1):
           pixel = img[r,c]
           hi[pixel] += 1
   return hi
```

```
def generateHistogram2(flattenImg):
   hi= [0]*256
   for pixel in flattenImg:
       hi[pixel] += 1
   return hi
```

I used 2 different methods to evaluate the results. I simply print the histogram array with print function. Then I visualized the results using plt.plot.

```
[134, 55, 24, 8, 7, 1, 4, 2, 3, 0, 3, 0, 2, 1, 2, 1, 1, 1, 3, 2, 0, 3,
0, 2, 0, 2, 2, 3, 1, 2, 1, 0, 4, 1, 2, 3, 2, 0, 1, 0, 1, 2, 4, 2, 0, 4,
0, 1, 0, 1, 3, 2, 1, 0, 2, 1, 1, 1, 2, 0, 2, 1, 2, 3, 1, 0, 2, 1, 2, 1,
2, 3, 1, 4, 4, 3, 4, 6, 6, 4, 11, 24, 26, 15, 13, 14, 18, 7, 13, 19,
13, 14, 23, 23, 15, 12, 19, 20, 21, 22, 16, 30, 22, 30, 27, 31, 33, 48,
50, 58, 50, 35, 61, 63, 59, 89, 89, 89, 94, 84, 96, 93, 94, 120, 144,
131, 145, 145, 165, 174, 187, 199, 238, 256, 293, 277, 303, 351, 422,
376, 417, 477, 441, 503, 451, 506, 500, 449, 488, 504, 580, 574, 603,
667, 730, 699, 674, 753, 771, 832, 945, 1013, 1183, 1314, 1302, 1098,
1108, 1107, 1035, 1044, 1032, 1014, 806, 775, 667, 597, 702, 736, 766,
946, 861, 1114, 1152, 1229, 1294, 995, 438, 165, 58, 44, 26, 20, 12,
16, 13, 8, 6, 8, 2, 2, 2, 3, 0, 1, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0,
0, 0, 0, 0, 0, 2, 2, 1, 1, 2, 1, 1, 1, 1, 1, 0, 1, 0, 0, 1, 0, 1, 1, 2,
0, 0, 1, 0, 3, 1, 0, 1, 1, 1, 1, 3, 11, 36, 64, 155]
```



2.6. Equalize Histogram Function

2.6.1. Cumulative Sum Function

Parameters

- values: the given list

Return Value

- my_list: list of cumulative sums

We are trying to produce a new list whose i^{th} element will be equal to the sum of the (i + 1) elements (cumulative sum). In our function, we created an empty list that name is my_list.

In the for loop we sum all previous elements for the current element. Then we add new values to our new list. We will use this function for equalizing histogram of an image.

```
def cumulative_sum(values):
   my_list=[]
   k=0
   length=len(values)

   for i in range(length):
       k+=values[i]
       my_list.append(k)

   return my_list
```

2.6.2. Normalization Function

Parameters

- entries: the given list

Return Value

- scaled: normalized values

In this function, we did a min-max normalization. We are taking a cumulative sum list of 1D array(flatten image). We subtracted the minimum element from every element and we multiplied the value by 255. Because we want to normalize our pixels between 0 and 255. Then, we calculate the maximum value minus the minimum value. After all these procedures, we apply this formula.

$$x_{scaled} = \frac{x - \min(x)}{\max(x) - \min(x)}$$

```
def normalization(entries):

   numerator = (entries - np.min(entries)) * 255
   max_min = np.max(entries) - np.min(entries)

   scaled = numerator / max_min
   return scaled
```

2.6.3. Equalize Histogram Function

Parameters

- img: image that we load using opencv

Return Value

- img_new: image that we improve contrast

First off all, we flatten our image. Then, we calculate the cumulative sum list. Then we apply the normalization function. After that, we convert the 1D array to 2D according to image shape.

```python
def equalizeHistogram(img):

   flatten_img = img.flatten()
   cumulativeSum = cumulative_sum(generateHistogram2(flatten_img))
   cumulativeSum_norm = normalization(cumulativeSum)

   img_new_his = cumulativeSum_norm[flatten_img]
   img_new = np.reshape(img_new_his, img.shape)

   return img_new
```