4.11.2018

Biometrics

Report 1

Dominik Gonciarz

How to run

GitHub repository https://github.com/AdPod/APBI contains a set of separate tools to manipulate images. Those are python implementation of provided algorithms which run command line.

To run, type in console $ python <filename> <image path>

All programs depend on python packages, listed in file requirements.txt

Implementation details

Grayscale Conversion:

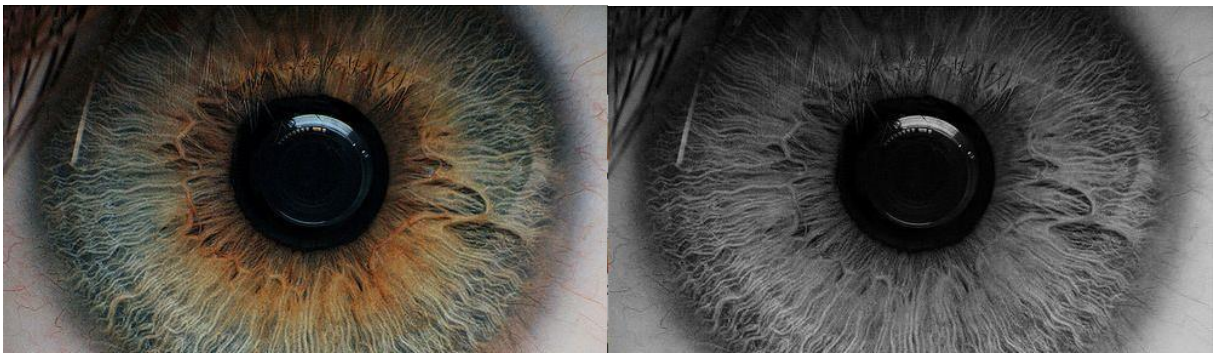
There are two main algorithms to convert color image (encoded as array of RGB tuples) to grayscale.

One can either calculate arithmetic mean of those three values or use different coefficients for different colors. Calculated value is the amount of white in grayscale image, 255 – white, 0 – black anything between will be some hue of gray. Images produced by second method gives closer approximation to color image because of characteristics of human eye.

The program loops through all pixels in the image and stores result in separate arrays for two algorithms:

```
for x in range(w):
    for y in range(h):
        r, g, b = arr[x, y]
        avg[x, y] = (r + g + b)/3
        wei[x, y] = 0.299*r + 0.587*g + 0.144*b
```

Example:



Color inversion

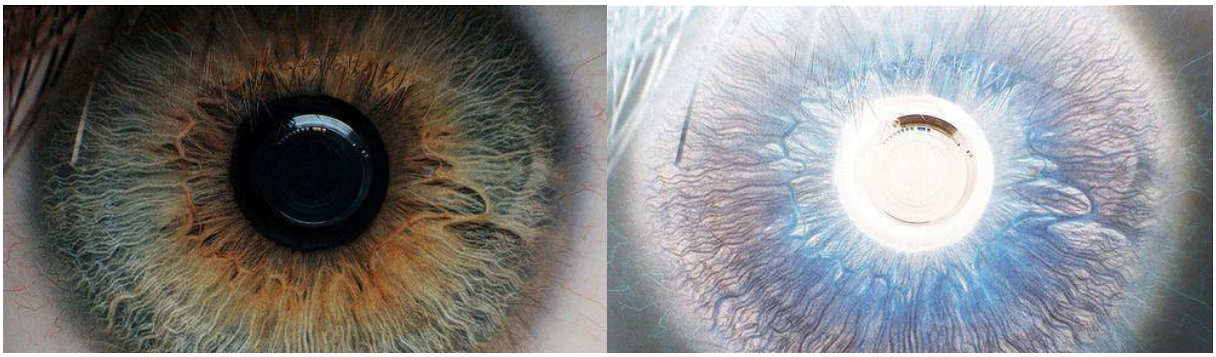
For each pixel in the image, algorithm separates color channels and each value is subtracted from maximal possible value of color intensity 255.

Dark blue (50,100,200) becomes some kind of dirty orange (205, 155, 55) etc.

Let python manage iterating through all value in array:

```
img – array of pixels
inv = bound(255 - img)
```

Example:



Brightness modification:

Modifying brightness of whole image is easy. Predefined and constant value is added to each channel in each pixel. Brightness modification will often produce values exceeding interval [0, 255], crucial is to limit numbers to proper colors.

```
bound() is used to limit numbers to the interval
img – array of pixels
m – modification coefficient
br = bound(img + m)
```

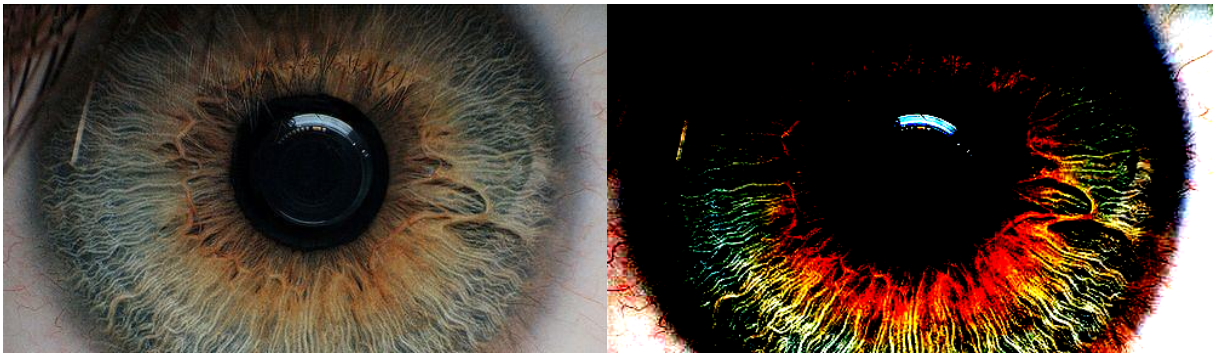
Examples:

Contrast enhancement

Increasing contrast will magnify differences between colors. The greater factor we apply (up to some level) the more intensive image we will receive.

```
factor – predefined value, here set for 8, can be changed
img – array of pixels
ce = bound(((img/255 - 0.5)*factor + 0.5)*255)
```

Example:



Threshold

Threshold reduces number of colors in the image to two. From grayscale picture we simply take all pixels with values lower than set threshold and replace them with black. Remaining are converted to white. Different values of threshold will give best results for different images.

```
wei - grayscale image stored as pixel array

thresholded = np.zeros_like(wei) # create array of zeros and the same shape
as wei

thresholded[wei > t] = 255 # whenever value of pixel in wei is > predefined
threshold t, set white in final array
```

Examples:

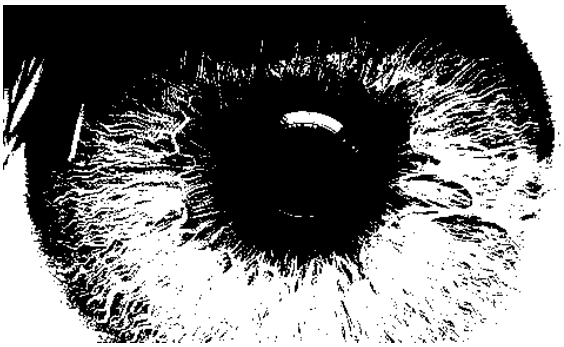
Horizontal + vertical projection

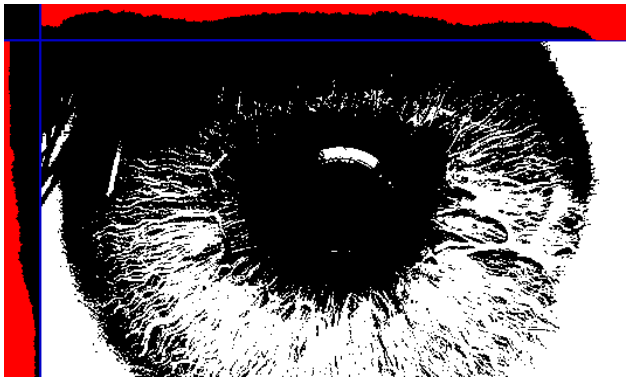
Algorithm counts number pixels of one color in binary image separately for rows and columns. Ratio for each measurement is visualized as red bar over or to the left.

Left and top padding has been added so graphs do not cover image.

```
thresholded.shape[0] - number of rows in array
for x in range(thresholded.shape[0]):
    sum_in_row = np.sum(thresholded[x, :])
    white_ratio = sum_in_row/thresholded.shape[1]/255
    stretched = int(round(30*white_ratio))
    for p in range(stretched, -1, -1): # place as many red pixels so it will match ratio of white/black
        projection[x + 32, p] = (255, 0, 0)
```

Examples:

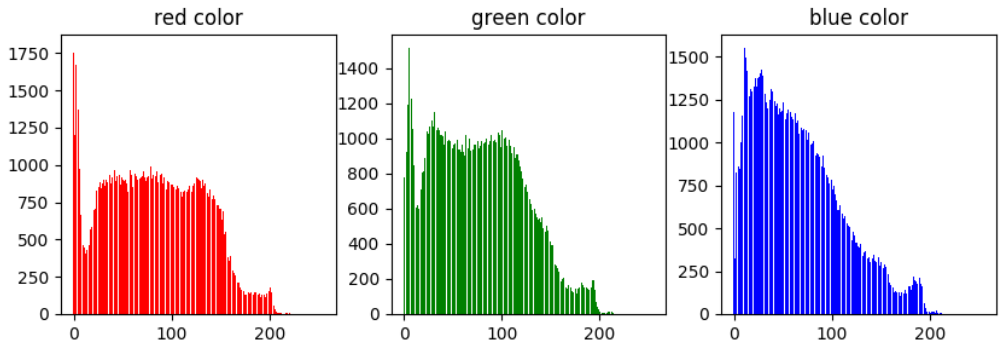




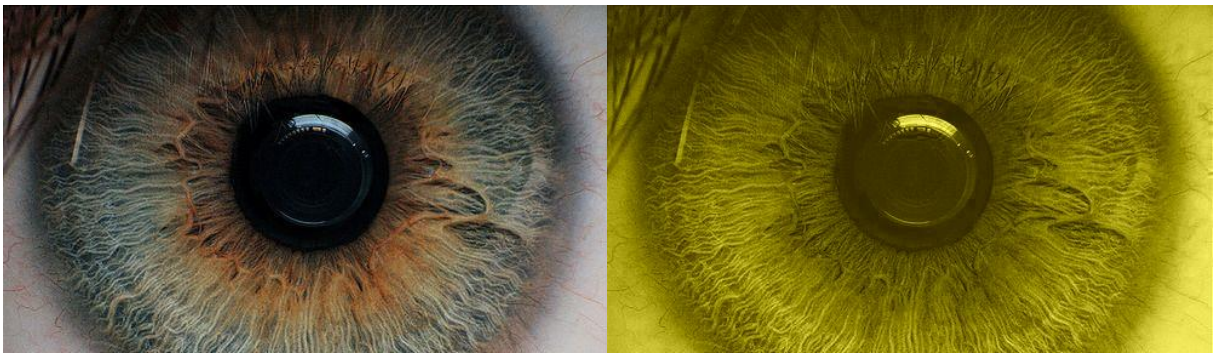
Histogram manipulation:

To perform histogram manipulations lut arrays are used. Those define how value of each pixel will be overwritten.

We start with drawing a histogram, following on is for eye photo used before.



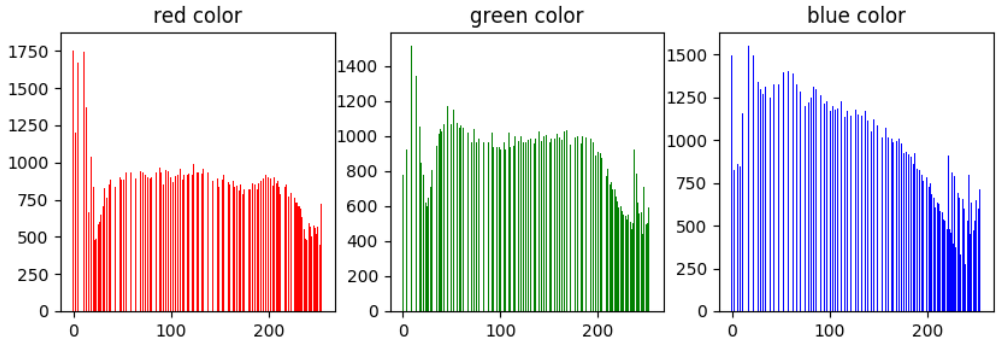
A) Predefined manipulation (R + 50, G + 40, B – 100)



```
lut – array of transformations for one channel (can be different for each color)

img – array of values for one channel

w, h = img.shape

flat = img.flatten()

for x in range(len(flat)):

    flat[x] = lut[flat[x]]

flat = flat.reshape((w, h))
```

B) Histogram equalization

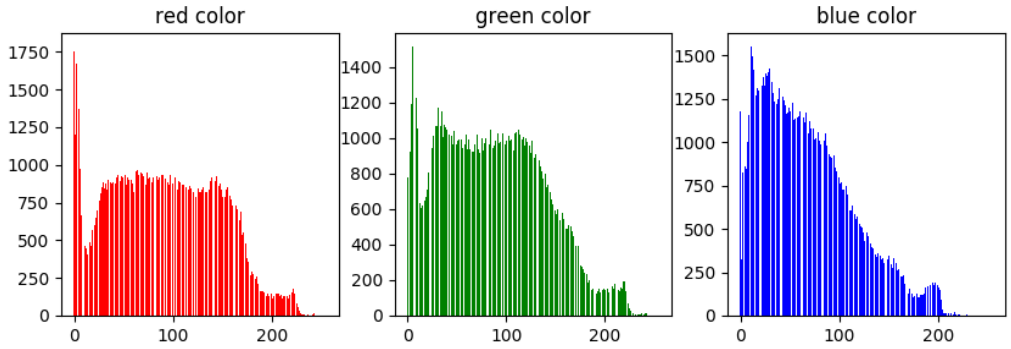
Histogram:




```
arr - array of values for one channel

occur_arr – function returning number of pixels having exactly this value of color, for one channel

lut_manip – function described above. Modifies array according to lut array

lut = np.zeros(256)

h, w = arr.shape

occur = occur_arr(arr)

d0 = occur[0]/h/w

const = 255/(1 - d0)

for i in range(255):

    lut[i] = (np.sum(occur[:i+1])/h/w - d0)*const

return lut_mainp(arr, lut)
```

C) Histogram expansion

Histogram:



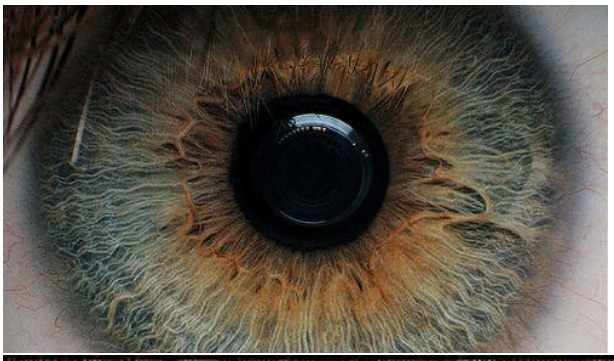
Filters:

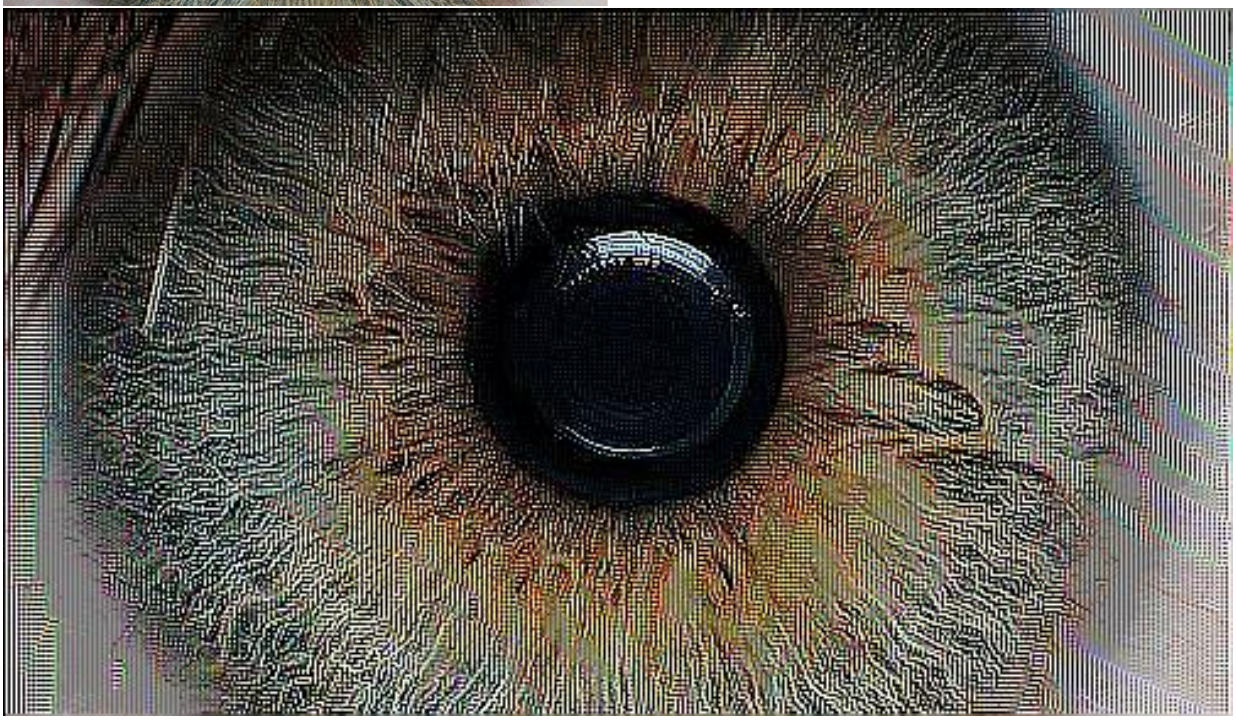
A) Averaging filter

B) Gauss filter

C) Sharpening filter

Here image was sharpened but also annoying chequered pattern was added.





D) Roberts Cross

E) Sobel filter

Eye image may look strange, but Sobel filter is aimed to detect edges, which eye has many. As we can observe edges on the fingerprint were correctly detected.