image processing code

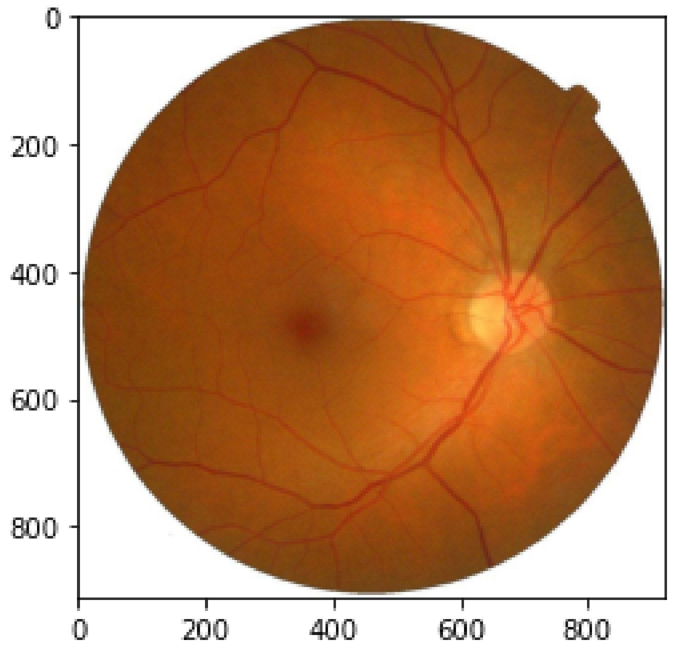
October 24, 2020

[1]:
```
import numpy as np
import skimage.io
import os
import matplotlib.pyplot as plt
from skimage import exposure
import numpy as np
import sys
```

[2]:
```
os.getcwd()
os.chdir('C:\\Users\\hp\\Desktop\\imagestoolbox\\')

Image=skimage.io.imread('retina.jpeg')
plt.imshow(Image,cmap='gray')
Image.shape
```

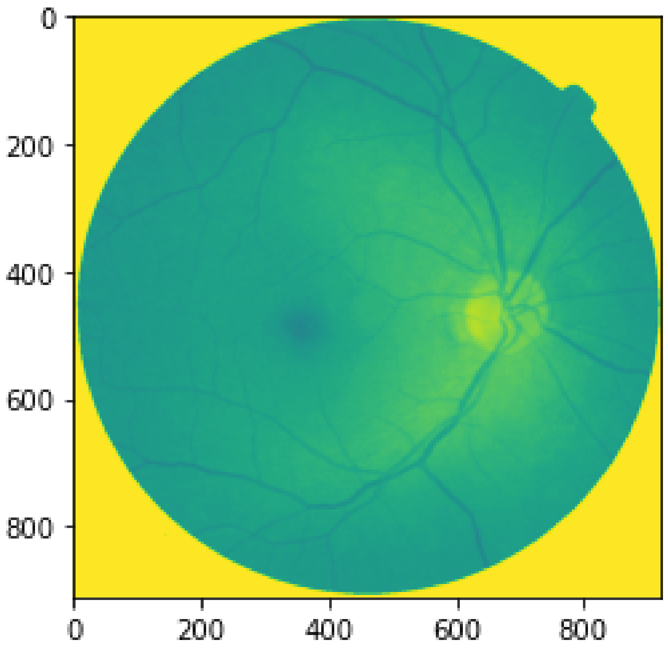
[2]:
```
(911, 922, 3)
```



Gamma correction, or often simply gamma, is a nonlinear operation used to encode and decode luminance in video or still image systems. A gamma value inf to 1 is sometimes called an encoding gamma, and the process of encoding with this compressive power-law nonlinearity is called gamma compression; conversely a gamma value sup to 1 is called a decoding gamma, and the application of the expansive power-law nonlinearity is called gamma expansion. Gamma correction function is a function that maps luminance levels to compensate the non-linear luminance effect of display devices (or sync it to human perceptive bias on brightness).

[3]:
```
def adjust_gamma(Image,gamma):
  I=skimage.color.rgb2gray(Image);
  I = I / np.max(I) ;
  I2= exposure.adjust_gamma(I, gamma);
  return I2
```

[4]:
```
I2=adjust_gamma(Image,0.5)
plt .imshow(I2);
plt .show();
```



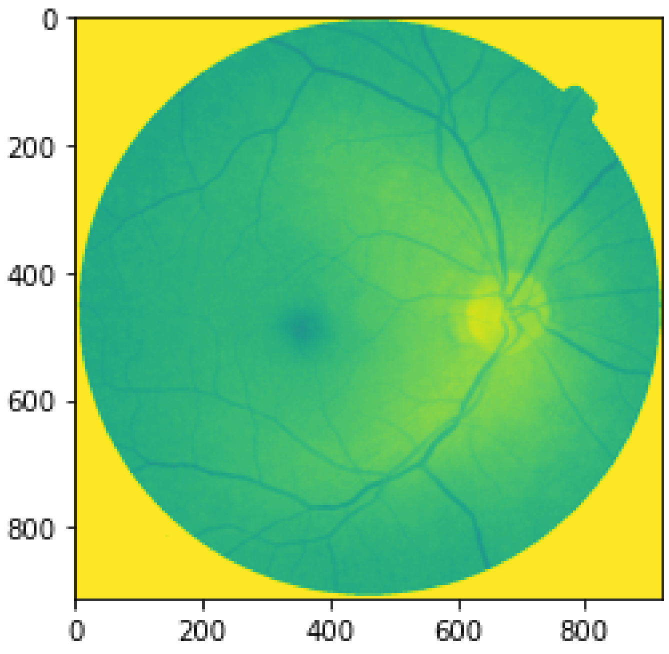
Contrast stretching (often called normalization) is a simple image enhancement technique that attempts to improve the contrast in an image by ‘stretching’ the range of intensity values it contains to span a desired range of values, e.g. the the full range of pixel values that the image type concerned allows.

[5]:
```
def contrast_stretching ( Image , E) :
   I=skimage.color.rgb2gray(Image);
   epsilon = sys .float_info . epsilon ;
```

```
    m = np.mean(I) ;
    I = I . astype("float") ;
    Ar = 1. / (1.+( m/(I+epsilon ) ) **E) ;
    return Ar;
```

[6]:
```
Ar=contrast_stretching(Image,0.8)

plt.imshow(Ar);
plt.show();
```



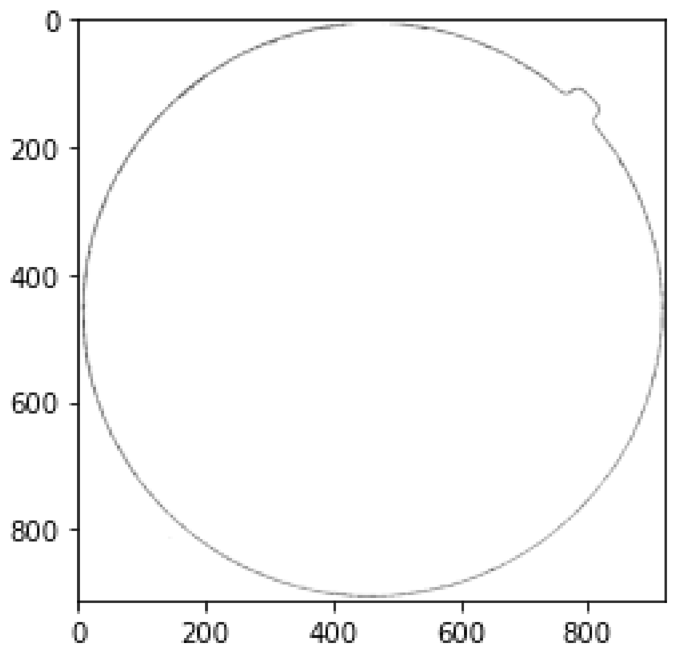
A Sauvola threshold is a local thresholding technique that is useful for images where the background is not uniform, especially for text recognition. Instead of calculating a single global threshold for the entire image, several thresholds are calculated for every pixel by using specific formulae that take into account the mean and standard deviation of the local neighborhood (defined by a window centered around the

[39]:
```
def Sauvola_Method(Image,Wind_size,R=128,k=0.5):
    imsize=Image.shape
    if(len(imsize)==3):
        Image=skimage.color.rgb2gray(Image)
        imsize=Image.shape
    Mask=np.zeros(imsize)
    nrows=imsize[0]
    ncols=imsize[1]

    #detrminig the mean and the sd for each pixel
    half_wind=Wind_size//2
    for i in range(half_wind,nrows):
        for j in range(half_wind,ncols):
            Window=Image[(i-half_wind):(i+half_wind),(j-half_wind):
↪(j+half_wind)]
            mean=np.mean(Window)
            std=np.std(Window)
            Mask[i][j]=mean*(1 + k * ((std / R) - 1))
    return Mask
```

[40]:
```
Mask=Sauvola_Method(Image,16)
I=skimage.color.rgb2gray(Image);
I=I>Mask
plt.imshow(I,cmap='gray')
```

[40]:
```
<matplotlib.image.AxesImage at 0x243d05c1ee0>
```



[ ]: