Assignment 2 – Basic Image Processing

This assignment will count towards your grade

Summary

The aim of this assignment is to work with the image processing techniques that are required as first step for almost all pattern recognition pipelines. You will perform histogram based technique (point operations) for thresholding, histogram stretching and perform basic matrix handling operation with numpy, scikit-image and matplotlib packages.

Introduction

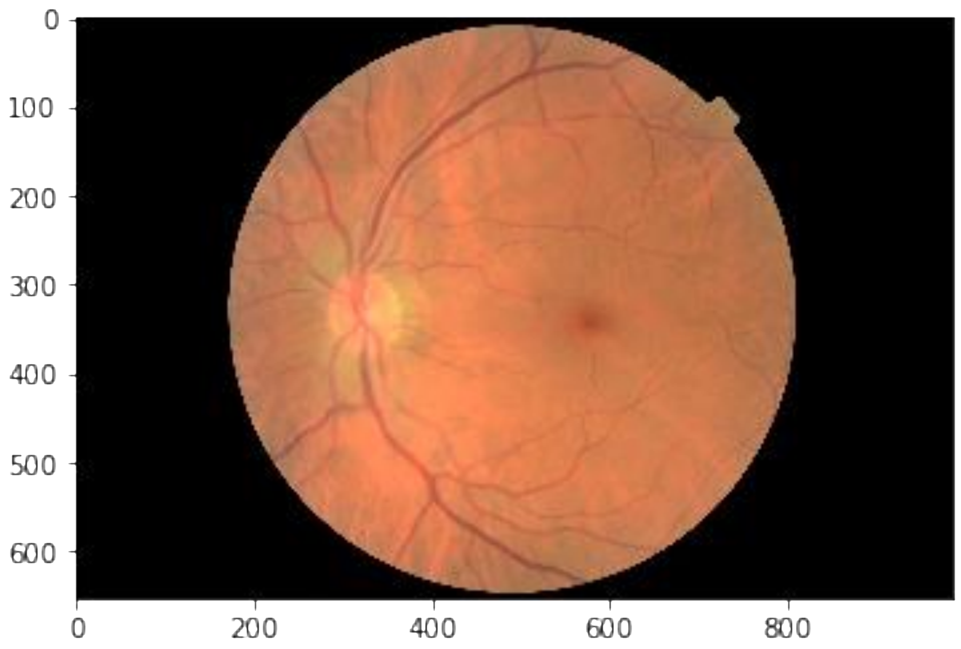
The following assignment needs to be presented as a Jupyter notebook. However it is suggested to write as much as possible of code as an external module and use Jupyter notebooks for visualization and code description only. This will allow you to easily debug the code.
Check the Resources/Suggestions section at the end of the document for help with the assignment.

Matrix Handling Operations (5 points)

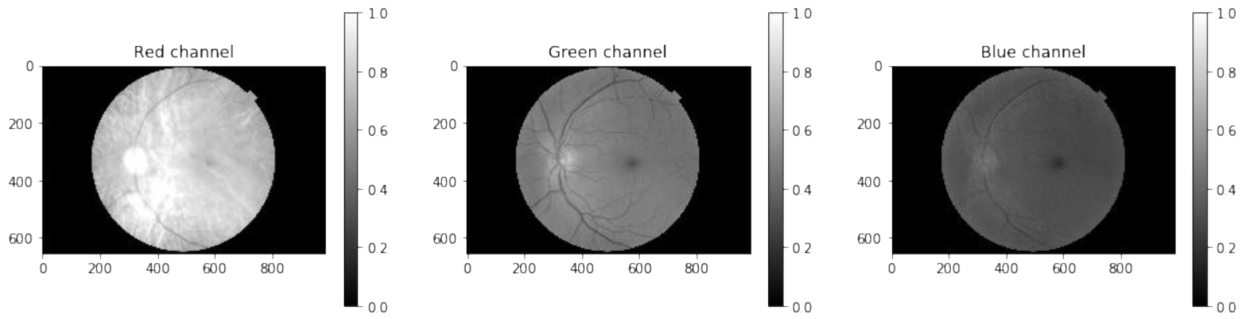
- Download the zip file containing the supporting data and decompress the content in a subdirectory separate directory
- Import the following packages at the beginning of your code:

```
import matplotlib.pyplot as plt
import numpy as np
import skimage.io as skio
import skimage.transform as sktr
import skimage.filters as skfl
import skimage.color as skcol
```

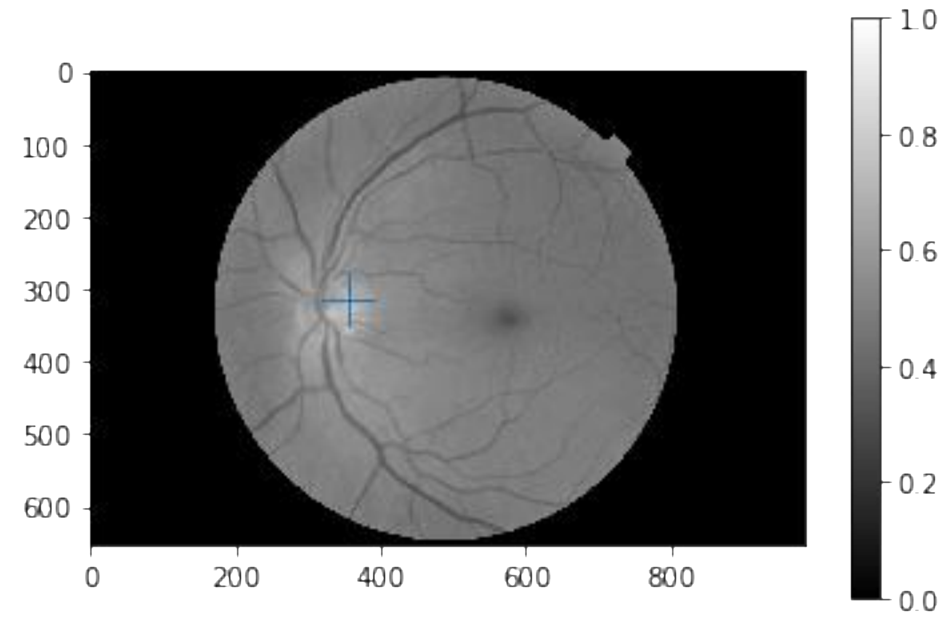
- Load retina image ‘data/68_left.jpeg’
  (check skio.imread function)
- Resize image to 20% of its initial size and display
  (check http://scikit-image.org/docs/dev/auto_examples/transform/plot_rescale.html)



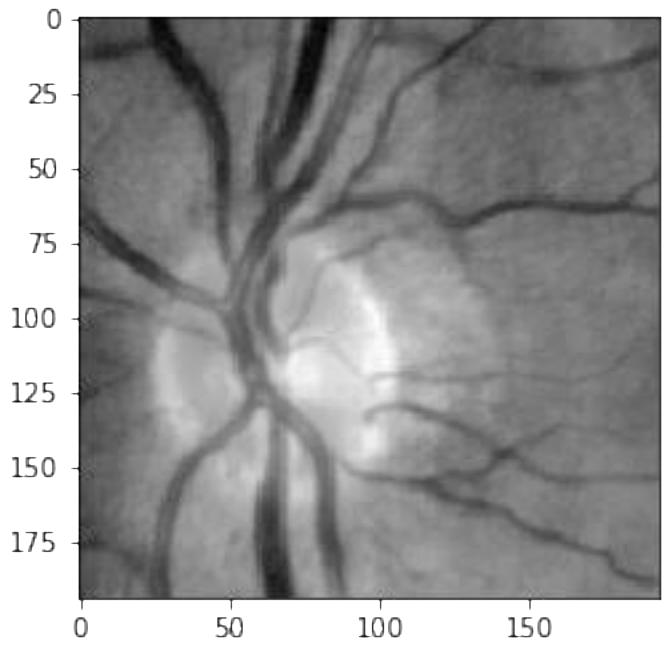
- Using array splicing separate the red, green and blue channel and display



- Convert the RGB image to grey level using the skcol.rgb2gray function and store it into a variable called imG
- In retina images the rough location of the optic nerve often corresponds to the brightest are of the image. Find the coordinates of the brightest point in imG.
    - You can use a nested for loops and if statements but you can also peform this task using these two functions np.argmax and np.unravel_index
    - Plot location with a cross on the image. You can perform this by first showing the image (using plt.imshow without calling plt.show), then plotting the cross (using plt.plot(x_coordinate, y_coordinate, '+', ms=20)) and finally calling plt.show(). If you are working from the console you will also have to call plt.hold(True) to avoid the immediate display of the image.

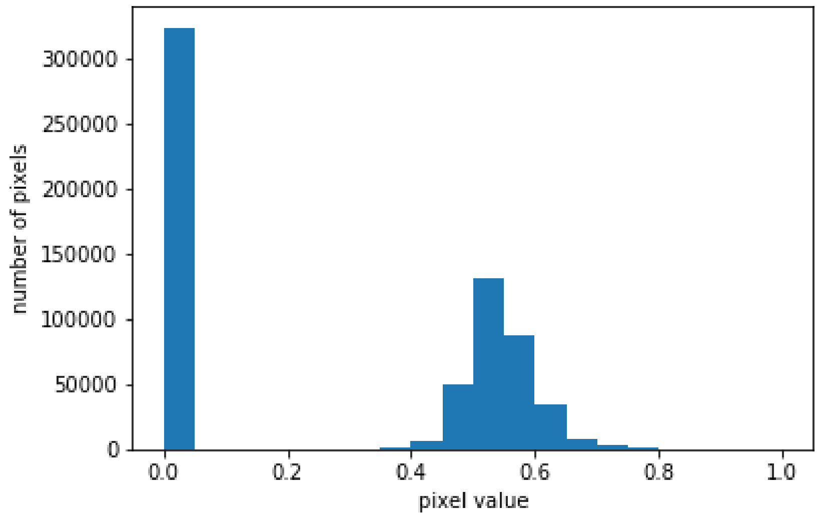


- Plot a window centered on the around the location of the cross. The size of the window should be 30% of the height of the image (the height of the image is the shape property). Use the array slicing operations to extract the window

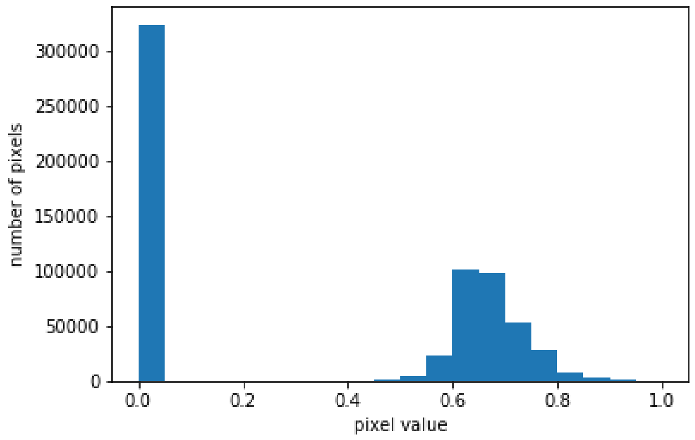


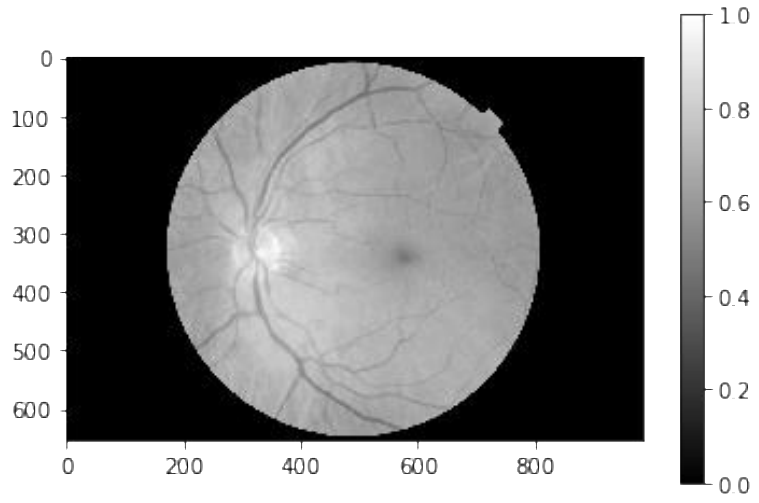
Histogram (5 points)

- Visualize histogram of imgG. Use the plt.hist function with 20 bins and a range from 0 to 1. Remember that you have to convert the image (represented as a 2D martrix) to an 1-dimensional array to compute the histogram correctly (check the flatten() function).

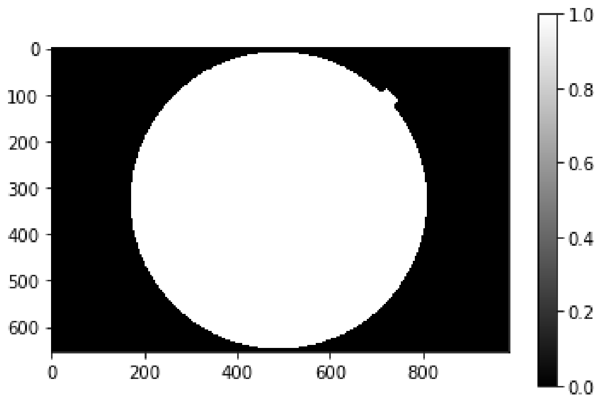


- Improve the image contrast by stretching the histogram from 0 to 1. To do so:
  - Find maximum and minimum pixel values (check max() and min() functions)
  - Stretch the histogram as discussed in class and at the following page http://homepages.inf.ed.ac.uk/rbf/HIPR2/stretch.htm
    Remember that the range of grey levels that we would like to stretch the histogram in this case is from 0 to 1
- Visualize stretched histogram and resulting image

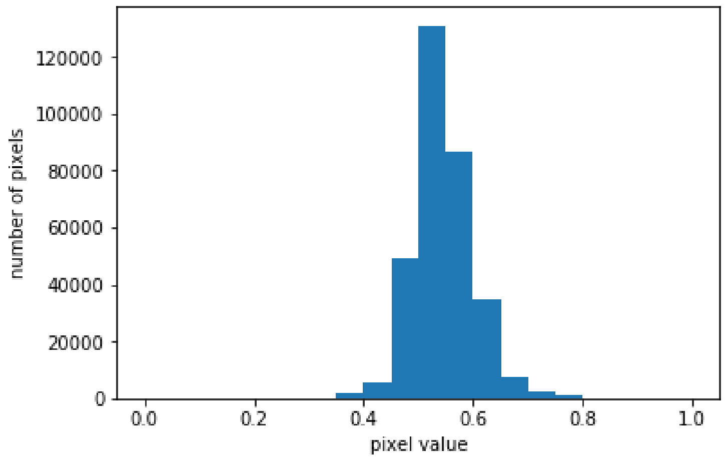




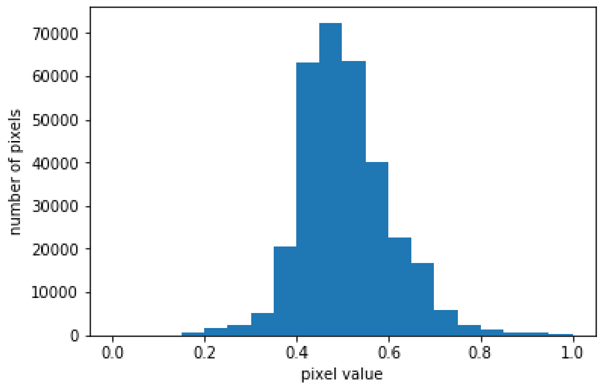
- Notice how the black area surrounding the retina image affects the result. We will now isolate the retina and stretch uniquely the histogram corresponding to it.
- Identify threshold splitting mask and image with Otsu method and print the result (check skimage.filters.threshold_otsu http://scikit-image.org/docs/dev/api/skimage.filters.html#skimage.filters.threshold_minimum)
- Create a boolean mask image by thresholding imG according to the value found by the Otsu algorithm (suggestion: check the “>” operator and http://scikit-image.org/docs/dev/user_guide/numpy_images.html)

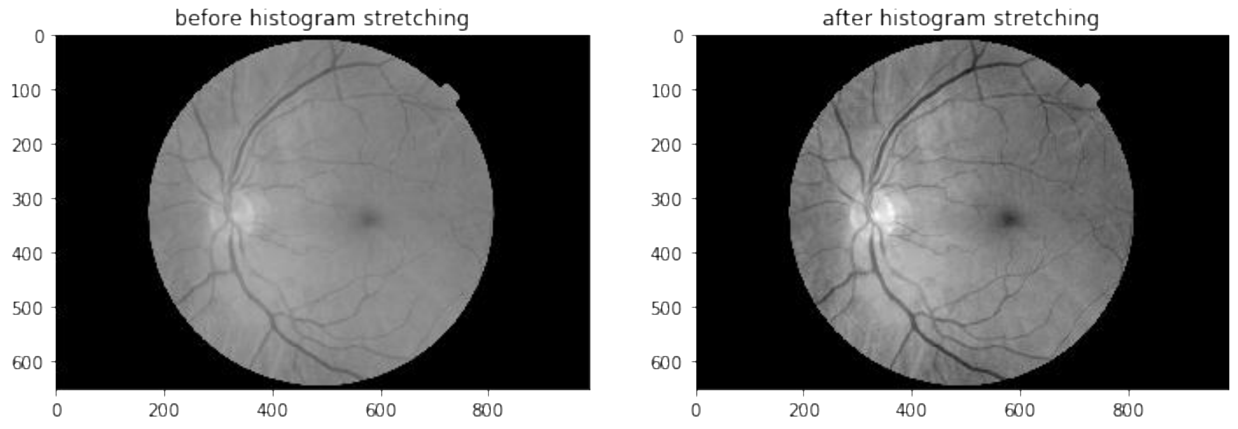


- Extract pixels inside the mask store them in a 1D vector and show the histogram (suggestion: remember that you can index a numpy array with a boolean mask thereby extracting only the values of interest, i.e. array[booleanMask]).



- Improve the image contrast by using histogram strechning according to the maximum and minimum values according to a 1D vector just created. Set all pixels below 0 to 0 and all above 1 to 1. Display the result. Compare with the previous histogram stretching approach.





- Create a function that given a file name perform the previous mask detection and histogram stretching. Apply it to ALL THREE images in the data directory and visualize the results.

Bonus Question (1 point)

Did the histogram stretching procedure does not work perfectly for all three images. What went wrong in one of them?

Resources/Suggestions

- The following document describe the image representation convention in scikit-image. This is a "must read". http://scikit-image.org/docs/dev/user_guide/numpy_images.html
- Remember the revise the reading for week 1 for more information about array/matrix slicing (http://www.scipy-lectures.org/intro/numpy/array_object.html)
- Matplotlib can have different behaviors depending on the type of "backend" used (see https://matplotlib.org/faq/usage_faq.html#what-is-a-backend). A trick to force a given backend that works well with the Pycharm debugger is by running this code at the beginning of the file

```
import matplotlib
matplotlib.use('TkAgg')
```

- This is an example of a function to display grey-level images with a colorbar in a given range.

```
def imShowBw( imgIn, immediateShow=True ):
    """
    Display an image its colorbar using a grey level color map.
    It forces a range between 0 and 1
    :param imgIn: 2D numpy array contating the image
    :param immediateShow: show the image immediately (i.e. with plt.show())
    :return: none
    """
    plt.figure()
    plt.imshow( imgIn, cmap=plt.gray(), vmin=0, vmax=1 )
    plt.colorbar()
    if immediateShow:
        plt.show()
```

Submission

- Now zip the folder containing your code and the notebook file as you did in assignment 01
- Convert the notebook to html and name it yourname-assignment02-nb.html.
- Compress original ipynb notebook file and supporting code and name it yourname-assignment02-src.zip
- Upload your notebook yourname-assignment02-nb.html and yourname-assignment02-src.zip to canvas before the deadline