CS 6643 - Computer Vision, Spring 2024 **Guido Gerig**

Project 1, Due Date: Feb 22, 2024 23:55

Out: Feb 1, 2024
Due: Feb 22, 2024 23:55

Late submissions: Late submissions result in 10% deduction for each day. The assignment will no longer be accepted 3 days after the deadline.
Grading: 25% for implementations and 75% for experiments and written reports including graphs, results, and critical discussion.

Office hours:

		Mon	Tue	Wed	Thur	Fri
Guido Gerig	gerig@nyu.edu				1400-1500 ZOOM	
Pragnavi Ravuluri Sai	pr2370@nyu.edu					800 - 1000
Sai Rajeev Koppuravuri	rk4305@nyu.edu			1200-1300 ZOOM		

Please remember that we also use **campuswire** for communication on homeworks.

Please read the following instructions carefully: We will check your code to make sure your implementation is your own, and it matches your results. **Your grade is primarily based on your written report.** This means going beyond just showing results, but also describing them. You should produce a standalone lab report, describing results in enough detail for someone else (outside of class) to follow. **Please read every page in this assignment.**

Please submit the report in PDF format and following are MUST included-

1. **It must have an index and contents page and all the questions, sub-questions must be indexed.**
2. **All the questions, sub-questions need to be addressed in the same order, design and implementation needs to be discussed in the report.**
3. **Results need to be shown in the report.**
4. **Must show the complete working code to all the questions at the end of the report.**

Optional:

- **Those who are willing to share the code notebook separately, make sure that all the cells are executed and it must have the desired output with proper implementation.**

A) Programming Questions

The purpose of this project is to get familiar with the coding environment for the rest of the course, reading/writing images, displaying images, and making graphs. You will implement histogram equalization.

Hint for implementation: You can assume an intensity range [0-255] and hardcode the size of the histogram to be 256. You can make sure you have an image of type uint8 with min 0 and max 255 with the python code:

A1) Histogram Equalization

Implement histogram equalization and apply to image **indoors.png**, which you will find uploaded on NYU Brightspace. The programming environment supported by our TAs is Python. You are encouraged to use libraries/built in functions to read/and write images and display figures and graphs, **but the rest of the implementation must be your own**. That specifically includes creating an image histogram (probability density function), computing the corresponding cumulative distribution function (CDF), and the creation of a new contrast adjusted image.

(Pre-Histogram Equalization).

(Post-Histogram Equalization).

Implement the following functions:

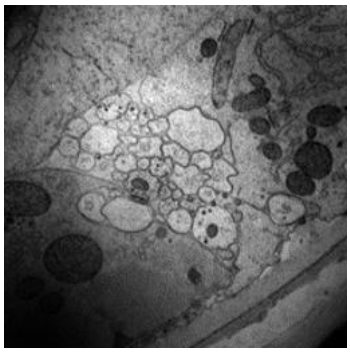
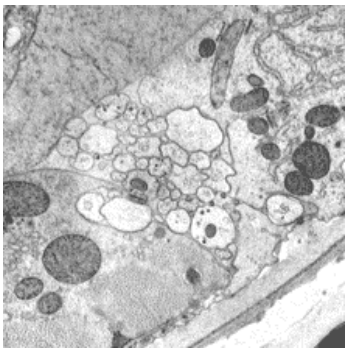
```
def create_pdf(im_in):
      # Create normalized intensity histogram from an input image
        return pdf

def create_cdf(pdf):
        # Create the cumulative distribution function from an input pdf
        return cdf

def histogram_equalization(im_in):
        pdf = create_pdf(im_in) # Your previously implemented function
        cdf = create_cdf(pdf) # Your previously implemented function
        # Create a histogram equalized image using your computed cdf
        return equalized_im
```

Write up a report including the following:

a. Include a brief introduction and description of how histogram equalization works.
b. Show **people.png** before and after histogram equalization, and the corresponding histograms (PDFs).
c. Discuss how the image and histogram have changed, and connect it back to your description in 1).
d. Show the cumulative distribution function before and after histogram equalization in a side-by-side figure. Describe what you see. Explain the shape of each CDF and relate it back to image contrast and intensity histogram shape.
e. Reapply the histogram equalization procedure on the corrected image. Show and discuss the results.
f. Apply histogram equalization to another low contrast image (greyscale). Show and discuss the results.
g. Include all code as an appendix, i.e. copy and paste all your code at the end of your report.

A2) Otsu Image Thresholding

For this section, you will implement **Otsu thresholding** algorithms to generate binarized images (where pixels are set to either 0 or 255 to produce a fully black and white image). See images **b2_a.png, b2_b.png,** and **b2_c.png** on NYU Brightspace alongside the assignment. **.** Please note that b2_b is an original microscopy image, and b2_c is the image corrected by multiplication with an illumination bias field as discussed in our first lecture (see slides). ***Show results for all of these images in your assignment***.

Implement the following functions:

```
def manual_threshold(im_in, threshold):
        # Threshold image with the threshold of your choice
        return manual_thresh_img

def otsu_threshold(im_in):
        # Create Otsu thresholded image
        return otsu_thresh_img
```

Write code to generate a **binary image** using a **manually chosen threshold**. Show the resulting binary image given a threshold of your choice, and add the threshold value to the report.

Otsu's Method is an algorithm to perform image thresholding with automatic threshold selection. The algorithm clusters image pixels into one of two possible classes by maximizing inter-class variance over the whole image. See https://en.wikipedia.org/wiki/Otsu%27s_method and Otsu.pdf on NYU Brightspace alongside the assignment for more details. Here, you will write code to implement Otsu's Method and threshold the same three images above.

a. Show the histograms for each image
b. Generate a plot of the inter-class variance as a function of the chosen threshold (i.e., x-axis with each possible threshold from 0-255, y-axis with the resulting variance)
c. State the inter-class variance of the image upon completion of the algorithm
d. Note the intensity threshold chosen by the algorithm
e. Show the resulting binary image produced by the algorithm
f. Discuss the results.
    i. Does the automatic threshold produce a decent result?
        - If yes: what elements of the image are separated from each other? Are there any improvements that could still be made?
        - If no: why does the algorithm fail to produce a decent result?

Include all code as an appendix, i.e. copy and paste all your code at the end of your report.

A3) Creative Part

Now that you have a code for histogram equalization, you can apply, extend and test the method to applications on your own. Choose the specific area of histogram matching and test the method on your own images

- Histogram matching: Instead of histogram equalization, we can choose a source image (to be improved) and a target image (with good contrast). Implement histogram matching (discussed on the last page of the hand-written notes on histogram analysis. Test how this works with a target image of similar scene content, or with an image from a completely different scene.

TODO

Follow the below instructions to preprocess your test images before applying Histogram Equalization and Thresholding methods mentioned in this project.

Preprocessing images clicked from a mobile phone:

```
# Load the image from the file location
# Rescale the image to desired dimensions (ideal 256x256)
# Convert the image into gray-scale numpy array
# Convert the image to type uint8 and scale intensity values to the range 0-255
# *Note* numpy min/max flatten the 2D array, so you obtain the min/max of the entire image
from PIL import Image
import numpy as np
original_image = Image.open('file_location')
resized_image = original_image.resize((256, 256))
im = np.array(resized_image.convert('L'))
im_uint8 = ((im - np.min(im)) * (1/(np.max(im) - np.min(im)) * 255)).astype('uint8')
# *Note* im_uint8 is the normalized grayscale image
```