

Project 1

Out: Sept. 14, 2023**Due:** Oct. 5, 2023 (deadline: 11:55PM)

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 report including graphs, results, and critical discussion.

Office hours:

		Mon	Tue	Wed	Thur	Fri
Guido Gerig	gerig@nyu.edu				2-3pm ZOOM	
Pragnavi Ravuluri Sai	pr2370@nyu.edu	4-6pm				
Sai Rajeev Koppuravuri	rk4305@nyu.edu			4-5pm		

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

Please read the instructions carefully. Note: we will not be running code. Rather, 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 a single PDF/HTML with all code included as an appendix.**

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.

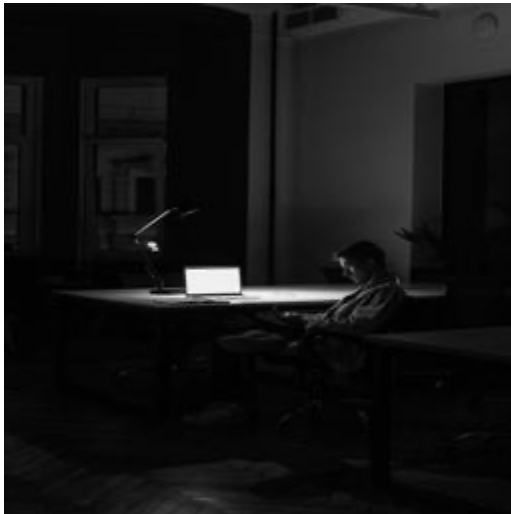
Hint for implementation: You can assume an intensity range [0-255] and hardcode the size of the histogram to be 256. You can make sure you have an image of type uint8 with min 0 and max 255 with the python code:

Project 1

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
```

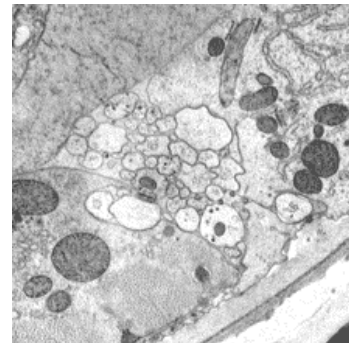
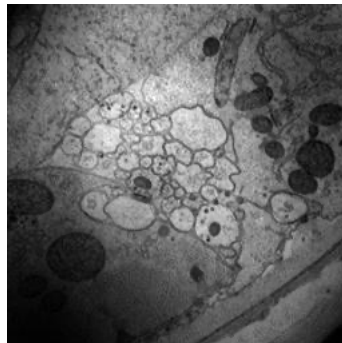
Project 1

Write up a report including the following:

- Include a brief introduction and description of how histogram equalization works.
- Show **indoors.png** before and after histogram equalization, and the corresponding histograms (PDFs).
- Discuss how the image and histogram have changed, and connect it back to your description in 1).
- 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.
- Reapply the histogram equalization procedure on the corrected image. Show and discuss the results.
- Apply histogram equalization to another low contrast image (greyscale). Show and discuss the results.
- 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
```

Project 1

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?
 1. If yes: what elements of the image are separated from each other? Are there any improvements that could still be made?
 2. 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 and Otsu thresholding, you can apply and test methods to applications on your own. Just choose one specific area to extend and test. A few possibilities that you may consider:

- 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.
- Extension of histogram equalization or histogram matching to color RGB images.
- Extension of Otsu thresholding with one threshold to a segmentation via two thresholds, resulting in an image with 0/1/2 labels for 3 regions.
- Others of your choice

Project 1

Appendix

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
```