CS 6643 - Computer Vision, Spring 2019 **Guido Gerig**

Assignment 1 - Practical

Out: 2 / 12 / 2019
Due: 2 / 28 / 2019 (deadline: midnight)

Submission Instructions:

a) **What to submit:**
   i) A .zip file containing your **code** along with **all of the generated graphs and images.**
   ii) A **SEPARATE PDF** report with all graphs and images. Also discuss in the report what are the results you are getting and why do you think results are that way.
b) **Where to submit:** NYU Classes

Late submissions: Late submissions result in 10% deduction for each day. The assignment will no longer be accepted 3 days after the deadline.

Office hours:

		Mon	**Tue**	**Wed**
Guido Gerig	Office 10.094	2-4pm		
Andrew Dempsey	ad4338@nyu.edu	10-12am		
Anshul Sharma	as10950@nyu.edu		10-12am	
Bhavana Ramakrishna	br1525@nyu.edu			10-12am

Location: cubicle spaces in 2 Metrotech, 10.098 A, B, D, E, H

B) Programming Questions

B1) Histogram and CDF

Write code that reads the provided 2D color image as input and performs the following: (you can find the raw image **b1.png** uploaded on NYU classes alongside this assignment)

Assignment 1 - Practical

1. Generate and show intensity histograms for each **color channel** in the provided image (red, green, blue)

2. **Discuss**: does the shape of each color channel histogram reflect the visible properties of the image?

3. Convert the color image into a grayscale image using the **luminosity method**, where the final gray pixel $I = (0.3 * R) + (0.59 * G) + (0.11 * B)$. Display the generated image (See https://www.tutorialspoint.com/dip/grayscale_to_rgb_conversion.htm)

4. Generate and show the histogram for your newly converted **grayscale** image

5. **Discuss**: does the shape of the grayscale histogram reflect the visible properties of the image? How does it compare with the color channel histograms generated above?

6. Normalize the histogram by the image size to create a **probability density function (pdf)** and plot the function

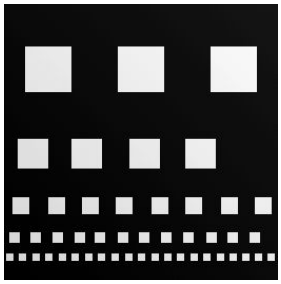
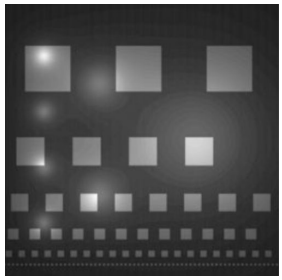
7. Calculate the **cumulative distribution function (CDF)** from your PDF and plot the function.

8. Write a function that implements the **histogram equalization** algorithm described in the book. Show both the **resulting image** and the **plot of the equalized histogram** after applying your function to the given grayscale image.

9. **Discuss**: how does the histogram equalization process affect the appearance of the image? Is the resulting histogram completely flat / uniform? Why or why not?

B2) Image Thresholding

For this section, you will implement **image thresholding** algorithms to generate binarized images (where pixels are set to either 0 or 255 if they are above or below a certain threshold to produce a fully black and white image). See images **b2_a.png, b2_b.png,** and **b2_c.png** on NYU classes alongside the assignment. ***Show results for all of these images in your assignment.***

1. Write code to generate a **binary image** using a **manually chosen threshold**. Show the resulting binary image, and note the threshold you chose

2. **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 classes 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?