Introduction

This assignment addresses some of the topics we covered in class. There are two parts to the assignment:

1. Becoming familiar with reading, manipulating, displaying and writing images to file, as well as using plotting functions. More specifically the assignment also covers working on various image thresholding methods.

2. Manipulating the pixels of color image and playing with color spaces

Download the homework1.zip file from myCourses `Contents`→`Programming_assignments`→`Homework 1`; this contains the instructions and images needed for the assignment.

Requirements

You should perform this assignment using Python and openCV, and it is due on **Sunday September 23rd by 11:59pm**. You are required to submit your code and a report containing short write-ups of what you did along with graphics displaying the results you obtained. You are strongly encouraged to start the assignment early. You are welcome to ask questions and have discussions about the homework on myCourses discussion but please do not post your solutions or any closely related material. If there are parts of the assignment that are not clear to you, or if you come across an error or bug please don't hesitate to contact the TA or the Instructor. Chances are that other students are also encountering similar issues. According to the RIT academic integrity policy, you are not allowed to share your solutions or post them to any publicly available location.

You are allowed to collaborate with other students as far as discussing ideas and possible solutions. However you are required to code the solution yourself. Copying others' code and changing all the variable names is not permitted! For this assignment, you are not allowed to use solutions from similar assignments in courses from other institutions, or those found elsewhere on the web. If you access such solutions YOU MUST refer to them in your submission write-up (penalties may apply). Your solutions should be uploaded to myCourses via Dropbox before the deadline.

Your submitted zipped file for this assignment should be named **LastnameFirstname_hw1.zip**. Failure to follow this naming convention will result in delays in getting your grade. Your zipped file should contain: (i) a PDF file named LastnameFirstname_hw1.pdf with your report, showing output images for each part of the assignment and explanatory text, where appropriate; (ii) the source code used to generate the solutions (with code comments). Please make sure your code compiles and runs successfully before submitting it. We should be able to execute your code for each part of the assignment in turn. Include a `readme` file if necessary (especially if using several external libraries), to run your code.

Problem 1. Basic image manipulations (40 out of 50 points)

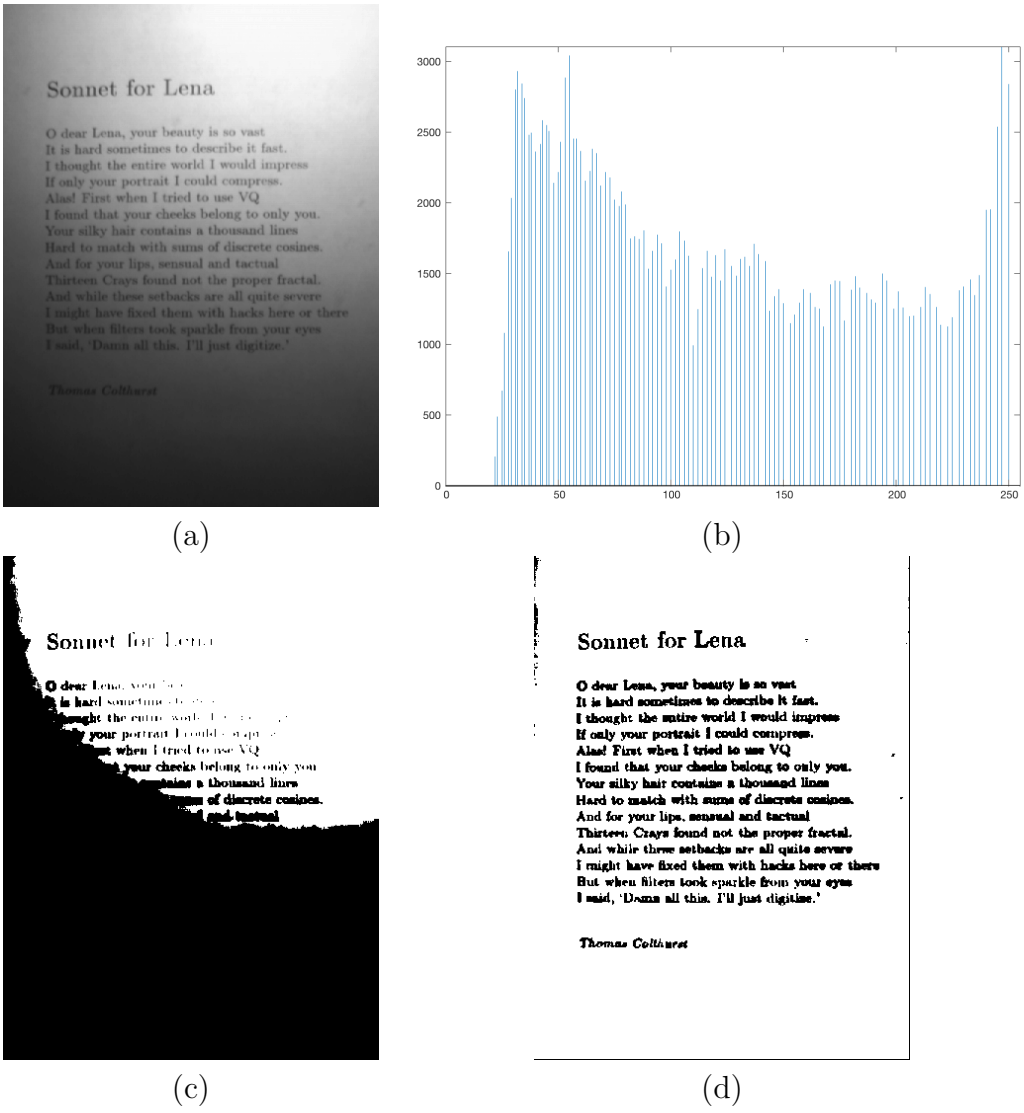

Figure 1: (a) the original image; (b) the histogram of the original; (c) the results after using my best guess at a global threshold value and (d) my best result using adaptive thresholding.

The goal of this problem is to get you to become familiar with basic image manipulations, including reading in images, writing out their modified versions, generating image histograms and running various types of image thresholding techniques. For this part of the assignment, from the folder `images`, you are to read in the image *sonnet.png* shown in Figure 1a:

(a) Plot its histogram, as shown in Figure 1b and use this to manually determine a value to use as the global threshold.

(b) Traverse through all the pixels in the image and set the value to 255 if greater than your threshold, else 0. Display your newly modified black and white image. Include your result in your report as in Figure 1c

(c) Now, implement adaptive thresholding, a localized version of what you performed above, using any one of the following formulas as we discussed in class. Turn in the version that gives the best looking results on your image. Include your best result in your report as shown in Figure 1d. The three options are:

(i) $t = \text{mean}(N \times N) + C$

(ii) $t = \text{median}(N \times N) + C$

(iii) $t = ((\max - \min)/2) + C$

Note: You will have to find the appropriate values of the window size $N \times N$ and the constant C. I used trial-and-error but you might be able to find a better way. But remember that these values change based on which threshold finding technique you use.

Your submission should contain the all the code files used to address this problem. We will run the code on other similar test images. Your report should contain a short writeup to include which threshold finding approach you used, graphics of the results you obtained, similar to Figure 1 and what values of N and C worked best under those circumstances. Extra credit will be given for any additionally creative solutions and/or any insights shared about how the problem was solved.

Problem 2. Playing with Color (60 points)

Figure 2: (a) the original image; (b) Alice-no-blue; (c) the results after uniformly adding a constant value to the image; (d) the results after adding more color to the image.

For this first assignment the goal is for you to get to familiar with working with different types of images, grayscale and color images. You get to practice more image manipulations, transforming pixels, breaking stuff, etc., it should be fun!
For this second problem, we will be working with the Disney® poster of the movie *Alice-in-Wonderland* located in the folder `images`. For this part of the assignment, you are to read in the image *Alice.jpg* shown in Figure 2a. Note that unlike the previous problem, this is a 'jpg' and not a 'png' file. Your tasks for this problem include:

(a) Getting and setting pixels: Since this is a color image, it will have three channels - Red, Green, Blue - after you read it in. Find out which one is the blue channel and

set all its values to zero to get an image which we call "Alice-no-blue" as in Figure 2b. Display your results in your report.

(b) Converting color to grayscale: Next, we want a grayscale rendition of this poster, but we want the best method to do this. So,from the folder `images`, load in the color swatch image *color_swatch.jpg*. Convert it to grayscale using an equally weighted mean, i.e. $I = \frac{(R+G+B)}{3}$ at each pixel. Now try a weighted sum which approximates the calculation of a phenomenon known as *relative luminance*; this approximation is given by $I' = 0.299R + 0.587G + .114B$ again at each pixel. In your report, show the original swatch along with the results of the two transformations. Discuss the effects of both conversions on the swatch and how perceptually meaningful each transformation is.

(c) Shifting the image colors: Write a function `shiftIm` that reads in an image, an image channel (0,1 or 2) and a constant value k. The function should add the constant value to the existing pixel values in that channel. The resulting image is not very useful since the values are not easily readable by the image library.

(d) Hence, write another function `clampIm` that takes an image and a maximum value m and ensures that if the value in each channel exceeds m, the value is clamped at$m$. Similarly, the minimum value is clamped at 0 if it was originally below 0. Now you can view the results of your clamped shifted image. Figure 2d shows the results of adding 120 (or 0.45) to each of the three channels equally. Discuss the effects of this function on the input image. What could this functionality be useful for?

(e) Now we will translate colorspaces - from RGB to HSV (hue, saturation, value) and back. Hue can be thought of as the base color of a pixel. Saturation is the intensity of the color compared to white (the least saturated color). The Value is the perception of brightness of a pixel compared to black. Use the inbuilt openCV function to convert your the original RGB *Alice* poster to HSV. To adjust the coloring of the original image, slightly increase the saturation in the HSV space by adding a value say between 0.10 and 0.25, to the S channel (use your `shiftIm` and `clampIm` functions, for the S channel). Convert back to RGB to view your modified image. Increase/decrease your value to get the best looking colorization of *Alice*. Display the final result in your report and discuss your findings.

As in Problem 1, your submission should contain the all the code files used to address this problem. We will run the code on other test images. Your report should contain the responses to the questions/discussions/images solicited in the write-up of the problem.
This assignment is due on **Sunday September 23rd by 11:59pm**.