ECE558 Homework 01

Due 09/06/2018

How to submit your solutions: put your report (word or pdf) and results images (please use .png format) in a folder named [your_unityid]_hw01 (e.g., twu19_hw01), and then compress it as a zip file (e.g., twu19_hw01.zip). Submit the zip file through moodle.

Objective: Knowing images and Getting started with Matlab or **Python (recommended)** manipulating images (including how to read, save and show an image, and how to access and iterate pixels and to apply some basic operations, etc). It's your choice to select one language to work with. Please check the course website for some useful coding resources (http://www4.ncsu.edu/~twu19/teaching_posts/ECE558-DIS/) and if you are not familiar with those please go through the provided short tutorial material while using the official documents as a reference guide.

Problem 1. Print the provided *pictureMe.pdf* and figure out what is in the image by painting it, where the number in each cell represents the grey level pixel values (0 for black, and 255 for white). Use your pencil to paint the picture based on pixel values and turn in the photo of your final painting (named as pictureMe.png). *Please try it manually and do NOT use image show tools in either Matlab or Python. Have fun.*

Problem 2: Find your NCSU unity id (digit signature) in an image. Use the provided *wolves.png* as the running example.

1.1) Read and show the image, and then capture the screenshot of the window showing the image (with the snipping tool in your operating system, e.g., on Mac, you can use Command+Shift+4 to start the snipping function). Save the captured screenshot as *[your_unityid]_screenshot.png* (e.g., twu19_screenshot.png)

Write down the code you wrote for reading and showing the image in your report and prove the saved screenshot image.

1.2) Find the digit signature of your unity id.

Firstly, convert each non-digit character in your unity id to ASCII value (e.g., ‘twu19’ corresponds to 116, 119, 117, 19, both Matlab and Python have built-in functions for it and you can google those if you do not know yet).
Secondly, count the number of occurrence of each of the digit number in each color channel of the image (the number of occurrence could be zero, but definitely less than the number of pixels). You will need to know how to access/iterate all the pixels in an image.
Note that the image read function could have different ordering of color channels, e.g., Matlab uses RGB, but OpenCV uses BGR.

In your report, write down your code and summarize your results using a table similar to:

Character	ASCII	# in Red channel	# in Green channel	# in Blue channel
‘t’	116	?	?	?
‘w’	119	?	?	?
‘u’	117	?	?	?
19	19	?	?	?

Thirdly, change to 255 the pixel values of the 5 by 5 sub-image (if valid) centered at each occurrence and then show the result image.
For example, suppose an input image is of size (150, 150) and we use 0-based index (note that Matlab use 1-based index). Locations near image boundary need special cares. Assume ‘t’ was matched at two pixel locations, one is at (100, 101) in the red-channel, and the other at (1, 148) in the blue-channel. For the first occurrence, the valid sub-image will be located by (98:102, 99:103) and the red-channel values of the total 25 pixels will be changed to 255. For the second occurrence, the valid sub-image will be located by (0:3, 146:149) and the blue-channel values of the total 16 pxiesl will be changed to 255.
After you changed all the occurrence, save the result image as [your_unityid]_signature.png (e.g., twu19_signature.png)

If the occurrence is empty, it is fine as long as your code can handle this type of situations correctly.