CPS843/CP8307 Introduction to Computer Vision
Assignment 0 - MATLAB Warm-up
Due February 2nd, 2021 by 11:59am
(Worth 5% of final mark)

This assignment is designed to make sure you can load an image, manipulate the values, produce some output; see MATLAB tutorials on the course webpage (and search the web) for additional help. Use BrightSpace to submit your assignment. This assignment is to be done **individually**. Try to avoid using loops, as much as possible. Traditionally, loops are notoriously slow, due to MATLAB being an interpreted language. Instead try to use vectorized operations by applying a single MATLAB command (i.e., precompiled code) to an entire array. To get documentation for a particular MATLAB function, type `help` and the then command name. Finally, make sure you do the appropriate typecasting (i.e., `uint8` and `double`) when working with images.

You should submit a zip file that contains the following:

- the images (e.g., JPEG or some other easy to recognize format) and
- and your MATLAB files for the assignment, including "`a0_script.m`"; see the comments in "`a0_script.m`" for additional details.

Your assignment must run fully by invoking "`a0_script.m`".

1. Input images

   (a) Find two interesting images to use. They should be colour. You can find some classic vision examples at `http://sipi.usc.edu/database/database.php?volume=misc`. Make sure they are not larger than 512×512.

   Output: Display both images; see `imshow`() (**Tip**: Always make sure to set the display range of `imshow`() to a reasonable range for display, such as the minimum and maximum values of the image).

2. Colour planes

   (a) Load image 1 and store it in the variable `img`; see `imread`().

   (b) Swap the red and blue channels of image 1

   Output: Display new image

   (c) Create a monochrome image (call it `img_g`) by selecting the green channel of image 1

   Output: Display new image

   (d) Create a monochrome image (call it `img_r`) by selecting the red channel of image 1

   Output: Display new image

(e) Convert the image to grayscale; see `rgb2gray`()

Output: Display new image

3. Replacement of pixels

(a) Take the square of 100×100 pixels located in the centre of image 1 (grayscale version) and insert them into image 2 (grayscale version).

Output: Display new image

4. Arithmetic and geometric operations

(a) What is the minimum and maximum of the pixel values of `img_g`? What is the mean? What is the standard deviation? How did you compute these?

(b) Subtract the mean from all the pixels, then divide by the standard deviation, then multiply by 10 (if your image is zero to 255) or by 0.05 (if your image ranges from 0.0 to 1.0). Now add the mean back in.

Output: Display new image

(c) Shift `img_g` to the left by 2 pixels.

Output: Display new image

(d) Subtract the shifted version of `img_g` from the original. (**Tip**: Whenever performing arithmetic operations with images make sure you convert the image to a floating point type prior to usage, otherwise issues will arise since the default type is an unsigned integer.)

Output: Display new image

(e) Flip horizontally `img_g` from the original, i.e., flip image left-to-right.

Output: Display new image

(f) Change the intensities of `img_g` from the original, such that the lightest values appear dark and the darkest appear light.

Output: Display new image

5. Image noise

(a) Take the original colour image 1 and start adding Gaussian noise to the pixels in each colour channel. (**Hint:** Create a three-channel image containing Gaussian noise.) Increase the variance of the noise until the noise is somewhat visible; see `randn`(). To increase the noise variance, multiply the Gaussian noise output by the standard deviation you desire. What value for the standard deviation did you use?

Output: Display new image