COC202 Computer Vision
Lab 2 – Spatial domain image processing

In this lab, we will implement/use implementations of Matlab's image processing toolbox of some of the spatial image processing techniques that we came across in the lecture.

If you have not yet finished the exercises from the previous lab, do them first.

1. Write a function that is passed an image and returns its negative. Remember that Matlab is a matrix-based programming language and that consequently most image processing tasks can be accomplished **without** loops (i.e., without looping through every pixel of the image explicitly). Your function should hence be a rather short one.

   Download the *rose.tif* image from Learn and test your function on it.

   Matlab's image processing toolbox also contains an image negative (also known as image complement) function. Find it and verify that your function returns the same result (you can use `help images` to get an overview of the image processing toolbox).

2. Write a function that changes the image gamma. The function should be passed an image and a value for gamma, and return the modified image.

   Test your function on the *temple.tif* and *desert.tif* images from Learn to bring out the details in the highlighted and shadow areas of the images respectively.

   Hint: if you get an image that is almost completely white, then you forgot to scale the image to [0;1] prior to applying gamma. Have a look at the help for `im2double`.

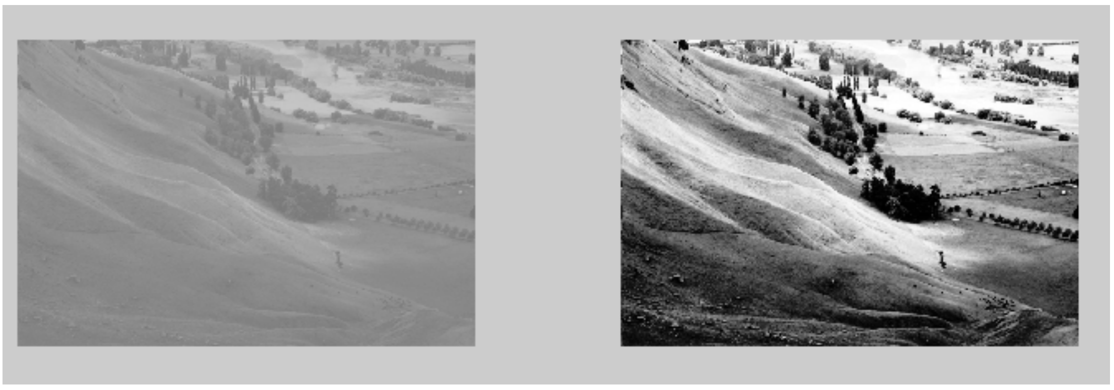
   Again, you should be able to find a function in the image processing toolbox that changes (among other functionality) the image gamma, so that you can compare your output.

3. Write a function that performs histogram equalisation for a (grayscale) image. The function should be passed an image and return the equalised image.

   Hint: the `imhist` function should be useful here.

   Test your function on the *landscape.tif* image from Learn.

   Not surprisingly, the image processing toolbox has a histogram equalisation function whose output you can compare to yours.



4. Matlab's image processing toolbox provides functionality for performing spatial image filtering with the `imfilter` and `fspecial` functions.

   Use the image processing toolbox to generate a 3x3 mean filter and confirm that its entries are the same as we discussed in the lecture.

   Now, use the mean filter to reduce the noise in the *rose_noisy.tif* image from Learn.

   Generate a Gaussian 3x3 filter and inspect its values.

   Then, use the Gaussian filter on the *rose_noisy.tif* image.

5. Does either of the filters generated in 5. remove the noise in the image? If not, why not?

   Identify an appropriate filter and use it on the image.

6. Generate a Laplacian filter and use it on the *rose.tif* image to perform edge detection.

   Hint 1: you will need to use 0 for alpha to obtain the filter from the lecture.
   Hint 2: you will need to convert the *rose* image to double so that you obtain the appropriate negative values in the output of the Laplacian.

   Then, use the filtered image to sharpen the original *rose.tif* image.

Once you have finished all exercises you may leave the lab.

Additional exercises for further study:

7. Download the *retim1.png* and *retim2.png* images from Learn. As you can see, one of them is of relatively low contrast and different colours compared to the other. We thus want to perform histogram matching to make the two more comparable.

   Write a function that is passed two greyscale images and returns the histogram matched image.

   Then use this function to perform histogram matching of the colour retina image (by performing histogram matching on all 3 colour channels).

   Matlab's histogram equalisation function also supports histogram matching, so once again you can compare your output.