ENGN6528 Computer Vision – S2, 2020
Computer Lab-1 (CLab1)

Objectives:

The goal of this lab is to help you become familiar with, and practice Python-based or Matlab-based basic image processing operations, binary image analysis, image filtering, template tracking and mathematical morphology.

This is CLab-1 for ENGN6528. The objective of this lab is to help you familiar with basic image I/O functions in either Matlab or Python. Note, in this lab, you are free to choose **any** of those two languages as you prefer and lab task descriptions are provided with both languages.

If you have not used Matlab or Python before, this lab is an opportunity to get you quickly familiar with basic language usages and relevant libraries for image processing and computer vision. Please note that Python is now increasingly used in computer vision, and we encourage you to practise it in this course.

Special Notes:

1. Each Computer Lab task lasts for three weeks and has two lab sessions: session-A and session-B in the first two weeks. Tutors/Lab instructors will provide basic supervision to both sessions. The third week has no lab, which is for you to complete and submit the lab report.
2. Your lab will be marked based on the overall quality of your lab report. The report is to be uploaded to Wattle site before the due time, which is usually on the Sunday evening of the third week after the announcing of computer lab tasks. (e.g. Sunday of Week4 for Clab1).
3. Your submission includes the lab report in PDF format as well as the lab code that generates the experimental result.
4. It is normal if you cannot finish all the tasks within the two 2-hour sessions -- these tasks are designed so that you will have to spend about 9 hours to finish all the tasks including finishing your lab report. This suggests that, before attending the second lab session (e.g. the lab in Week3 for Clab1), you must make sure that you have almost completed 80%.

Academic Integrity

You are expected to comply with the University Policy on Academic Integrity and Plagiarism. You are allowed to talk with / work with other students on lab and project assignments. You can share ideas but not code, you should submit your own work. Your course instructors reserve the right to determine an appropriate penalty based on the violation of academic integrity that occurs. Violations of the university policy can result in severe penalties.

C-Lab-1 Tasks

Task-1: Matlab (Python) Warm-up. (2 marks):

Describe (in words where appropriate) the result/function of each of the following commands of your preferred language in report. Please utilize the inbuilt *help()* command if you are unfamiliar with these functions.

Note: Different from Matlab, Python users need to import external libraries by themselves. And we assume you already know some common package abbreviations (e.g. *numpy = np*). [You only need to complete one set of questions either in matlab or python.] (0.2 marks each)

Matlab	**Python**
(1) a = [2, 4, 5; 5,2,200] ;	(1) a = np.array([[2,4,5],[5,2,200]])
(2) b = a(1,:);	(2) b = a[0,:]
(3) f = randn(500, 1);	(3) f = np.random.randn(500,1)
(4) g = f(find(f < 0)) ;	(4) g = f[f<0]
(5) x = zeros (1,100) + 0.55 ;	(5) x = np.zeros ((1,100)) + 0.55
(6) y = 0.4 .* ones(1,length(x));	(6) y = 0.4 * np.ones([1,len(x.transpose())])
(7) z = x - y;	(7) z = x - y
(8) a = [1: 200];	(8) a = np.linspace(1,200,200)
(9) b = a([end: -1:1]);	(9) b = a[: :-1]
(10) b(b <=50)=0;	(10) b[b <=50]=0

Hint: Do the necessary typecasting (uint8 and double) when processing and displaying the image data in the following tasks. For Python, please be aware of the default datatype of different libraries (e.g. Image, matplotlib, cv2). An improper datatype of an image will cause many troubles when you want to display the image.

Task-2: Basic Coding Practice (1 marks)

Write functions to process an input grayscale image with following requirements, where you need to write a script to load the given image in the Lab package, apply each transformation to the input, and display the results in a figure. For Matlab, you can use the *subplot()* function; for Python, we suggest using *matplotlib.pyplot.subplot()*. Please notice that each subplot needs to be labelled with an appropriate title. (0.2 marks each)

1. Load a color image, convert it to a grayscale image. Then map the grayscale image to its negative image, in which the lightest values appear darkest and vice versa. Display both grayscale images side by side.
2. Flip the image vertically (i.e, map it to an upside down version).
3. Load a colour image, swap the red and blue colour channels of the input.
4. Average the input image with its vertically flipped image (use typecasting).
5. Add a random value between [0,255] to every pixel in the grayscale image, then clip the new image to have a minimum value of 0 and a maximum value of 255.

Task-3: Basic Image I/O (2 marks)

In this task, you are asked to:

1. Take three frontal face photos of yourself, under different lighting conditions, against a white wall. (i.e., as if a passport photo). The image should be in landscape shape (i.e., the longer side is in the horizontal direction). (0 marks).
2. Re-scale the images to a size of 1024 columns x 720 rows, and save them to JPG image files named 'face-01-UId.jpg', 'face-02-UId.jpg' and 'face-03-UId.jpg' (replace UId with your uniID, the resized images need to be included in your submitted file). (0 marks).
   Hint: Make sure the three photos are under different, yet normal, lighting conditions. Try to avoid "extreme" lighting conditions (e.g., pure dark, or pure white or saturated), otherwise you will add some unnecessary difficulties if you use these images in subsequent CLabs. (0 marks).
   Note: Your photos will be used for this class only, and for the C-Labs component of the course during this semester. All photo files uploaded to Wattle will be deleted at the end of this semester. By submitting your photos to Wattle, you agree you understand this. As they are your own photos of yourself, you are allowed to post your face photos to the public domain.
3. Choose one face image, for example, face-01, and then program a short computer code that does the following tasks:
   a. Read this face image from its JPG file, and resize the image to 720 x 480 in columns x rows (0.2 marks).
   b. Convert the colour image into three grayscale channels, i.e., R, G, B images, and display each of the three channel grayscale images separately (0.2 marks for each channel, 0.6 marks in total).
   c. Compute the histograms for each of the grayscale images, and display the 3 histograms (0.2 marks for each histogram, 0.6 marks in total).
   d. Apply histogram equalisation to the resized image and its three grayscale channels, and then display the 4 histogram equalization images (0.15 marks for each histogram, 0.6 marks in total).
      Hint: you can use inbuilt functions for implementing histogram equalisation. e.g. *histeq()* in Matlab or *cv2.equalizeHist()* in Python.

Task-4: Image Denoising via a Gaussian Filter (5 marks)

In this task, you are asked to:

1. Read in one of your colourful face images. Crop a square image region corresponding to the central facial part of the image, resize it to 256x256, and save this square region to a new grayscale image. Please display the two images. Make sure the pixel value range of the grayscale image is within [0, 255] (0.5 marks).
2. Add Gaussian noise to this new 256x256 image (Review how you generate random numbers in Task-1). Use Gaussian noise with zero mean, and standard deviation of 15 (0.5 marks).

Hint: Make sure your input image range is within [0, 255]. Kindly, you may need *np.random.randn()* in Python. While Matlab provides a convenient function *imnoise()*. Please check the default setting of these inbuilt functions.

3. Display the two histograms side by side, one before adding the noise and one after adding the noise (0.25 marks for each histogram, 0.5 marks in total).
4. Implement your own Matlab function that performs a 5x5 Gaussian filtering (1.5 marks). Your function interface is:
   my_Gauss_filter()

   input: noisy_image, my 5x5 gausskernel

   output: output_image
5. Apply your Gaussian filter to the above noisy image, and display the smoothed images and visually check their noise-removal effects (0.5 marks in total).
   One of the key parameters to choose for the task of image filtering is the standard deviation of your Gaussian filter. You may need to test and compare different Gaussian kernels with different standard deviations (1.0 marks).
   Note: In doing this task you **MUST NOT** use any Matlab's (or Python's) inbuilt image filtering functions (e.g. *imfilter(), filter2()* in Matlab, or *cv2.filter2D()* in Python). In other words, you are required to code your own 2D filtering code, based on the original mathematical definition for 2D convolution. However, you are **allowed** to generate a 5x5 sized Gaussian kernel with inbuilt functions.
6. Compare your result with that by Matlab's inbuilt 5x5 Gaussian filter (e.g. *filter2(), imfilter()* in Matlab, or conveniently *cv2.GaussianBlur()* in Python,). Please show that the two results are nearly identical (0.5 marks).
   Further reading material: http://setosa.io/ev/image-kernels/

Task -5: Implement your own 3x3 Sobel filter in Matlab or Python (3 marks)

Sobel edge detector.
You need to implement your own 3x3 Sobel filter. Again, you **must not** use an inbuilt edge detection filter. The implementation of the filter should be clearly presented with your code.

Test it on one of your face images (we suggest converting it to gray-scale) (1 marks), and compare your result with inbuilt Sobel edge detection function (0.5 marks). Briefly explain how the results of the Sobel filter connect to the gradient of the image and discuss this filtering in relation to implementation efficiency (1.5 marks).

Task-6: Image Rotation (3 marks)

Choose one of your favorite images (among your face photos) and resize it to 512 x 512,

1. Implement your own function *my_rotation()* for image rotation by any given angle between [-90^0, 90^0]. Display images rotated by -90^0, -45^0, 15^0, 45^0, and 90^0 (0.20 for each image, 1.0 mark in total).
   Note: positive for clockwise rotation.
2. Compare forward and backward mapping and analyze their differences (1 mark).
3. Discuss different geometric transformations (eg, rotation, translation, shear, affine, etc.) (1 mark).
   Hint: When analyzing the difference, you can focus on: (1) visual results; (2) the principles in terms of formulation or others relevant; (3) advantages and drawbacks; (4) the computational complexity. You can also think about it from other aspects.

Submission Quality (4 marks)

Successful submission, correctly submitting the whole lab package. (1 mark)
Report quality, including clear description, presentation and correct report format (e.g. table captions and reference). (2 marks)
Code efficiency, quality, including clear demonstration, comments. (1 mark)

============= END of C-Lab-1 ============