Lab 1: Image Transformations

NUS CS4243, Semester 1, 2020

TA: Junbin Xiao
Due on: 03-09-2020 23:59

Objective

The objective of this lab is to familiarize yourself with basic image processing and transformation operations. The lab has three parts:

1. Histogram Normalization & Equalization
2. Gaussian & Laplacian Image Pyramids
3. Image Blending

You are free to use any functions in `numpy` to help you, but other built-in functions for image processing are not allowed. If you are new to Python and Jupyter Notebooks, please review the materials in the Python Refresher in `LumiNUS>Files>Labs` to help you start your work.

Part 1: Histogram Equalization and Normalization (35%)

In this part, you are asked to implement the following 5 functions: `cs4243_resize()`, `cs4243_rgb2grey()`, `cs4243_histnorm()`, `cs4243_histequ()` and `cs4243_histmatch()`. The expected outcomes of each function are shown in in Fig. 1 and `main.ipynb`. The code skeletons for the functions are given in `transform.py`.

For simplicity, we consider only greyscale images in this lab. Given a colour image, first resize it to a smaller image and then convert it from RGB to greyscale. Afterwards, perform histogram normalization, equalization or matching to enhance the image contrast. Image normalization (equalization) is to stretch the range of image intensity linearly (uniformly). Histogram matching is to make the image histogram match with the histogram of the reference image, which means that you have to find a mapping of intensity between two images according to their cumulative histograms.

If you have forgotten how normalization or equalization works, please review the material from Lecture 2. You can also find the appended online resources to help you.

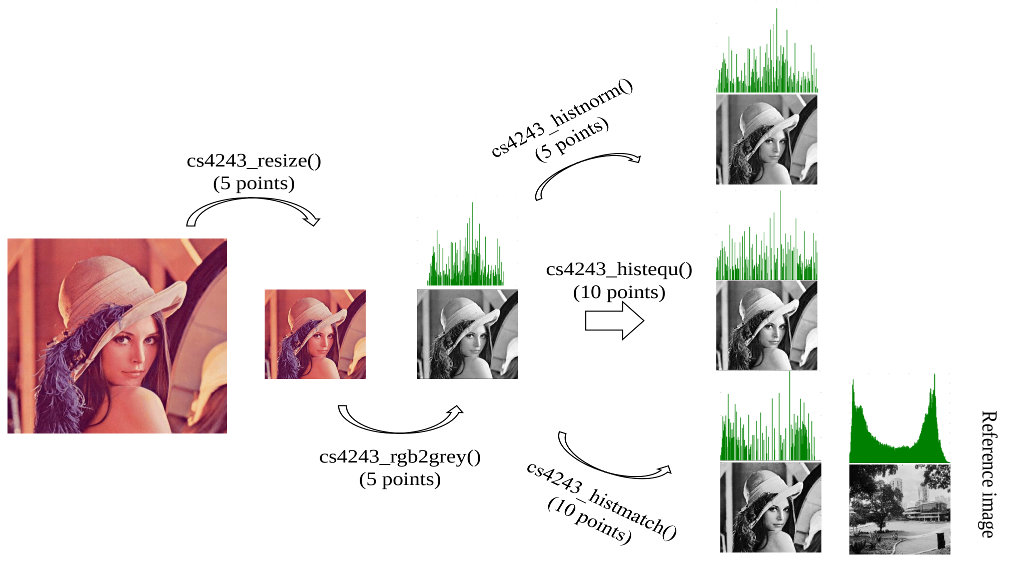

Figure 1: Image histogram normalization, equalization and matching.

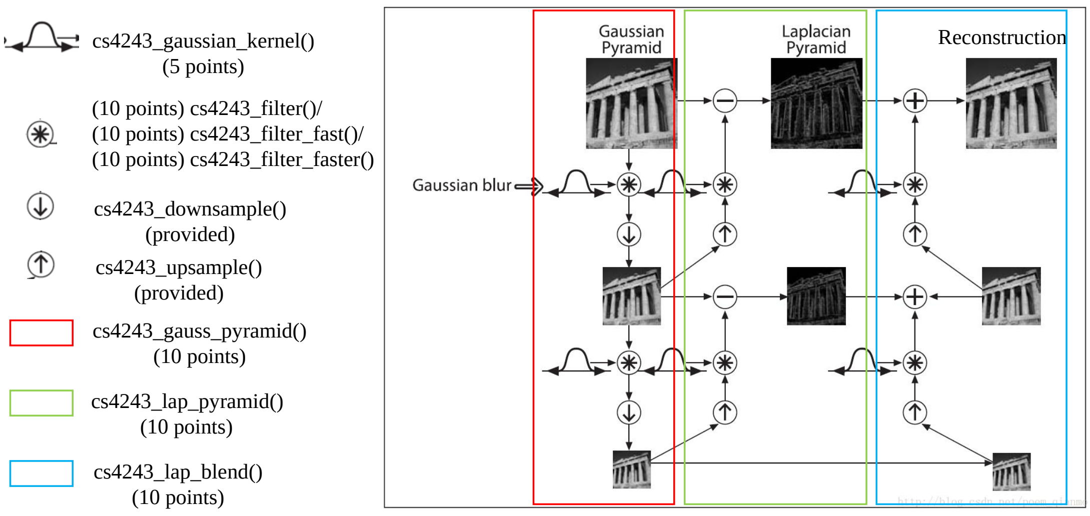

Figure 2: Gaussian & Laplacian Pyramid.

Part 2 Gaussian and Laplacian Pyramid (55%)

Fig. 2 illustrates the pipeline for constructing a Gaussian and Laplacian image pyramid as well as reconstructing the image from the Laplacian pyramid. Functions to be implemented are: `cs4243_gaussian_kernel()`, `cs4243_filter()`, `cs4243_filter_fast()`, `cs4243_filter_faster()`, `cs4243_gauss_pyramid()`, `cs4243_lap_pyramid()` and `cs4243_lap_blend()`. Aside from these functions, you may also need `cs4243_resize()` and `cs4243_rgb2grey()` to convert images to greyscale. As illustrated in Fig. 2, the kernel and filtering functions will be used for Gaussian blurring during pyramid construction. Additionally, 3 different filtering functions that work at different speeds are to be implemented (see `transform.py` for details).

Part 3 Image Blending (10%)

One neat use for Laplacian image pyramids is to blend two images together (see Fig. 3). For image blending, you separately build the Laplacian pyramids for the two constituent images and linearly combine them different pyramid levels during reconstruction.

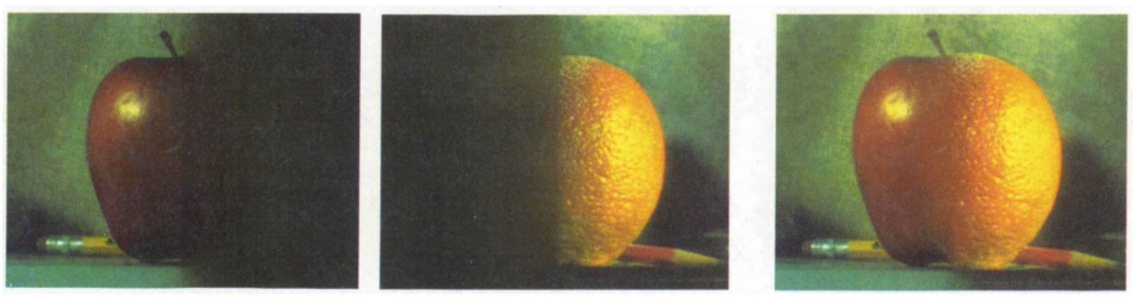

Figure 3: Apple and orange blending example.

The general approach for image blending is as follows:

1. Construct Laplacian pyramids L_A and L_B from composing images A and B.
2. Build a Gaussian pyramid G_R on a mask R which selects the pixels coming from A vs B
3. Construct a combined pyramid L_c from L_a and L_b by using the elements of G_R as weights, where $L_c(i,j) = G_R(i,j) * L_A(i,j) + (1 - GR(i,j)) * LB(i,j)$
4. Reconstruct belended image from L_c.

An example of linearly combining two images with a mask is given in `main.ipynb`.

Useful Resources and Extension Reading

Nearest-Neighbour Image Resizing [link]
Histogram Equalization[link]
Histogram Normalization[link]
Image Filtering[link]
Blending[link]

Hand in

Files to be submitted are `transform.py` and `main.ipynb`. Please zip them into a file named `AXXX1_AXXX2_AXXX3.zip`, where AXXX is the student number of the group members. Each group should submit only once. Note that we will use hold-out testing examples to test the functions in `transform.py`. Therefore, `transform.py` is essential and `main.ipynb` is for reference only in case you fail some testing cases. Groups with missing files or incorrectly formatted code that does not run will be penalized. The submission deadline is 03-09-2020 23:59. Q&A sessions for Lab 1 are held on 25/8/2020 & 28/8/2020.