NVLab Summer School 2023

HW1: Image Processing

TA:
林采熹 (cclin2189@gmail.com)
黃文嬿 (karenhuang0513@gmail.com)

Deadline: 2023 / 7 / 18 (二) at 23:59

Grading Policy:

1. In the programming assignment, the code and report (HW1_report.pdf) should be compressed into a ZIP file. Also, please write a README file to explain how to run your code and describe related characteristics used in your report. The report format is not limited.
2. You are required to finish this homework with Python 3. Moreover, built-in machine learning libraries or functions, e.g., *sklearn.linear_model*, are NOT allowed to be used.
3. Discussions are encouraged, but plagiarism is strictly prohibited (changing variable names, etc.). You can use any open source you want with a clear reference mentioned in your report. If there is any plagiarism, you will get 0 in this homework.

Submission:

Please follow the following format and naming rules when submitting files.

```
HW1_studentID_name.zip
    |----HW1_report.pdf
    |----README.md (or *.txt)
    |----Part1.ipynb, Part2.ipynb, Part3.ipynb (only *.ipynb)
```

- The zip file does not contain the dataset.
- You need to upload HW1_studentID_name.zip.

Programming:

In this problem, you may learn some basic image processing skills and get familiar with using PyTorch and cv2 in Python.

Data

We have prepared some files for you in the compressed folder. You can start your work by following the guidance in this instruction.

Please put the input images into resource/ and the processed images into output/. And you still need to show them in the ipynb.

Problem

Part 1. Color/Luminance Manipulation

A. Gamma Mapping:

Step1. Scale the pixel value from [0, 255] to [0, 1.0].

Step2. Apply the transformation:

$$O = I^{1/gamma}$$

Step3. Scale the pixel value back to [0, 255] for display.

B. Histogram Equalization:

Step1. Transform the color space from RGB to YCbCr or HSV.

Step2. Calculate the histogram of luminance.

Step3. Calculate the CDF.

Step4. Determine the look-up table based on CDF.

Step5. Update the pixel values.

Part 2. Image Filter

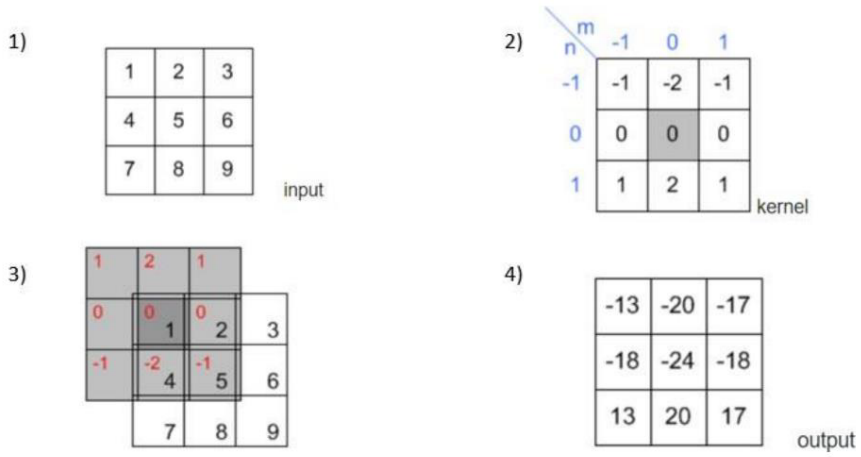
A. Convolution Implementation (FROM SCRATCH)

Built-in functions like cv2.filter2D() or numpy.convolve() are NOT allowed to use. You need to implement it using Numpy by yourself. We have created my_conv() for you, you need to finish the remaining part and you should call this function in Part 2B.

Step1. Determine your input image and a kernel with odd dimensions.

Step2. Do zero-padding to ensure that the output will not shrink.

Step3. Implement convolution.



B. Hybrid Image

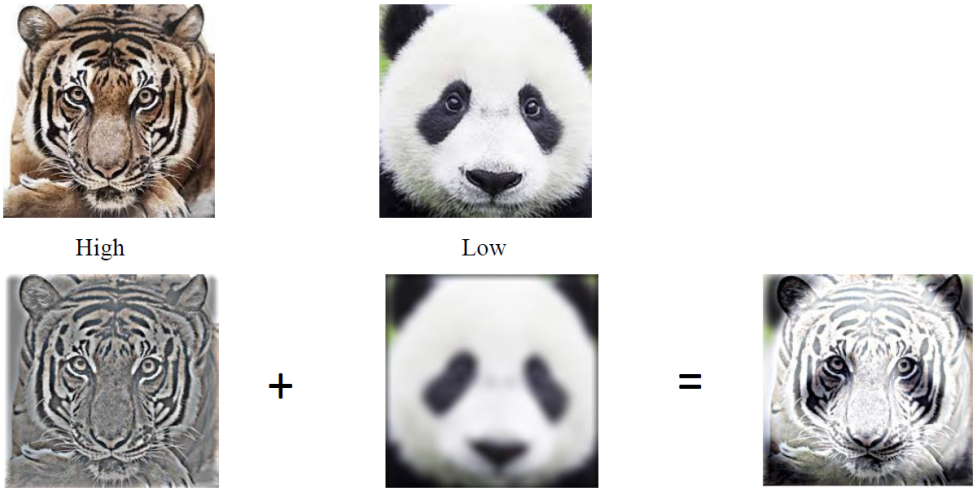
Hybrid images are static images that present different interpretation as the viewing distance changes. The basic idea is that high-frequency signal (e.g., edges, textures, etc.) tends to dominate perception when closely observing an object. However, from a distance, only the low-frequency (smooth) part of the signal can be seen. By blending the high-frequency portion of one image with the low-frequency portion of another, you get a hybrid image that leads to different interpretations at different distances.

Step1. Pick two appropriate images. Do some preprocessing if needed.
(Same size, and overlapping subjects)

Step2. Determine high-pass filter and low-pass filter. We have created a blur kernel for you (my_GaussianBlur()) to be the LPF. You can adjust kernel size and sigma to get better result.
(Hint: high pass filter → "subtraction")

Step3. Apply the two filters on different images then combine them.

For example:



Part 3. Albumentations

Albumentations is a Python library for fast and flexible image augmentations. Albumentations efficiently implements a rich variety of image transform operations that are optimized for performance, and does so while providing a concise, yet powerful image augmentation interface for different computer vision tasks, including object classification, segmentation, and detection.

In this part, you need to use **Albumentation** to do the image process (e.g. blur, rotation, flip, resize, etc.)

Part 4. Torchvision

Torchvision is a library for Computer Vision that goes hand in hand with PyTorch. It has utilities for efficient Image and Video transformations, some commonly used pre-trained models, and some datasets.

In this part, you need to use **Torchvision** to do the image process (e.g. blur, rotation, flip, resize, etc.)

Report:

1. (Part1.a) Compare the corrected results with gamma = 0.55, 1 and 2.2 .
2. (Part1.b) Show the original image and the processed output as well as their histograms.
3. (Part2.a) Use my_conv() to process the input image with different kernels (at least blur and sharpen).
4. (Part2.b) Try to use different parameters for my_GaussianBlur(), what do you observe?
5. (Part2.b) Try different pairs of pictures in the folder /resources or gather your own picture pairs , show your results of hybrid results and what you discover during the process.
6. (Part2.b) Describe what kind of information we acquire from two filters in Hybrid Image.
7. (Part2) What applications do you think this technology can be used for?
8. (Part3) Use **Albumentation** to do the image process (e.g. blur, rotation, flip, resize, etc.) (at least 3 kinds)
9. (Part4) Use **Torchvision** to do the image process (e.g. blur, rotation, flip, resize, etc.). (at least 3 kinds)
10. (Part3 and Part4) Discuss the difference of doing image processing using convolution techniques in Part2.a, using **Albumentation** and **Torchvision.**