ITE4052 Computer Vision (Spring 2025)
Programming Assignment 2

- Deadline: May 15th 11:59 PM.
- Submit the Python code (filename extension: .py) and a report summarizing the results (pdf) on the LMS.
- If you did this with an .ipynb extension (e.g., Colab or Jupyter Notebook), please convert it to a .py.
- You can use either English or Korean for the report.
- Late submission will get a half score.
- If you use ChatGPT, you can be caught for plagiarism (immediate F). Don't use it.
- Use the expression to calculate the PSNR score yourself, and write down the score in the report
- Provide a description of how each code was implemented in the report.

1. Mean Filtering [2pt]

Add a **Gaussian noise** (with any parameter) to your **own image** and apply the **Mean filtering** to the noisy image with kernel sizes of 3, 5, and 7, respectively. Compare the **PSNR** of the noisy image with that of the filtered images. Additionally, **briefly describe** the changes in image appearance after adding noise and applying filters, and compare the results across different kernel sizes with a focus on noticeable visual differences across regions. Don't use Python functions such as *skimage.util.random_noise ()*.

2. Unsharp Masking [1pt]

Perform unsharp masking with kernel sizes of 3, 5, and 7, respectively, for your **own image**. **Briefly describe** how each masked image differs from the original image, and compare the differences among the masked images with different kernel sizes, with a focus on noticeable visual differences across regions. Don't use Python functions such as *ImageEnhance.Sharpness()*.

3. Contrast Stretching [2pt]

Implement a code for (1) **Contrast stretching** and (2) **Gamma correction** with your own parameters for your **own image**. In the contrast stretching, experiment with multiple sets of $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ values, and in Gamma correction, test multiple $\boldsymbol{\gamma}$ values. Show the results and **briefly describe** how the outcomes change with different hyperparameter settings. Don't use Python functions such as (1) *ImageOps.autocontrast()*, *cv2.normalize()*, *cv2.equalizeHist()*, (2) *numpy.percentile()*.

4. Histogram Equalization [1pt]

Implement the **histogram equalization** code and apply it to your **own image**. Don't use Python internal functions such as *cv2.equalizeHist()*, *ImageOps.equalize()*. Show the **histogram** of the image **before and after the equalization**. **Briefly describe** the qualitative differences between the images before and after applying histogram equalization, focusing on noticeable visual differences across regions.

5. Image Upsampling [2pt]

Down-sample your test image by a factor of 4 (e.g., 256x256 to 64x64), and **display** its **up-sampled version** of the original size, using (1) **nearest neighbor**, (2) **bilinear**, and (3) **bicubic interpolation**. You can use Python functions for this problem. Compare the performance of the interpolation methods based on **PSNR**. Additionally, **briefly describe** the changes in image appearance after down-sampling and compare the results across different up-sampling methods with a focus on noticeable visual differences across regions.