E9 241 Digital Image Processing
Assignment 02

Due Date: September 25, 2024 - 11:59 pm **Total Marks:** 50

Instructions:
For all the questions, write your own functions. Use library functions for comparison only.

- Your function should take the specified parameters as inputs and output the specified results.
- Also provide the wrapper/demo code to run all your functions and obtain results. Your code should be self-contained i.e., one should be able to run your code as is without any modifications.
- For Python, if you use any libraries other than numpy, scipy, scikit-image, opencv, pillow, matplotlib, pandas, and default modules, please specify the library that needs to be installed to run your code.
- Along with your code, also submit a PDF with all the **results** (images or numbers) and **inferences** (very important: you may not be explicitly asked to give inferences in each question. You should always include your inferences from what you have observed). Include answers to subjective questions, if any.
- Put all your files (code files and a report PDF) into a **single zip file** and submit the zip file. Name the zip file with your name.

1. **Spatial Filtering and Binarisation:** Apply Gaussian blurring on the image `'moon_noisy.png'`. Generate a spatial Gaussian filter of size 41×41 by generating a filter kernel as:

$$G_f(x,y) = \frac{1}{K}\exp\left\{\frac{-(x^2+y^2)}{2\sigma_g^2}\right\}, \tag{1}$$

where $x, y \in \{-20, -19, \cdots, 19, 20\}$ and σ_g is the standard deviation of the Gaussian kernel. K normalizes the filter such that $\sum_x \sum_y G_f(x,y) = 1$. You can use a library function to convolve the input image with the kernel you created to obtain the Gaussian blurred image.

Apply Otsu's Binarization algorithm on the blurred image and note the optimal within-class variance ${\sigma_w^2}^*$ for a given σ_g blur parameter.

Plot the histogram and the binarized image for the blurred images and find their corresponding optimal within-class variances ${\sigma_w^2}^*$ for each $\sigma_g \in \{0, 0.1, 0.5, 1, 2.5, 5, 10, 20\}$ (0 corresponds to the input image itself). Find the optimal σ_g than minimizes ${\sigma_w^2}^*$. Comment on your observations.

(15 Marks)

2. **Fractional Scaling with Interpolation:**

   (a) Downsample the image `'flowers.png'` by 2 and upsample the result by 3 using bilinear interpolation.

   (b) Upsample the image by $3/2 = 1.5$ using bilinear interpolation.

   Observe what is different in both results and give comments.

   Hint: Use a zoomed-in patch of the image for a better visual comparison.

(15 Marks)

3. **Photoshop Feature:** Watch the video explaining the Brightness/Contrast feature in Adobe Photoshop (‘`photoshop_feature.mp4`’). Implement this based on your knowledge of pointwise operations applied to images.

   Implement two functions `brightnessAdjust(img,p)` and `contrastAdjust(img,p)`, that output the respective adjusted images where `p` controls the respective adjustments. In particular, for `brightnessAdjust`, `p=0` should output a black image, `p=1`, a white image, and `p=0.5` should output the input image itself without any adjustment. Output for `p` values between `0.5` to `0` should gradually change from the input to a black image. Similarly, for `p` values between `0.5` to `1`, the output should gradually from the input to a white image. For `contrastAdjust`, `p=0.5` should not alter the input image, `p=0` should output a grey image, and `p=1` should output a black and white image (would look like a binarized image). For all other intermediate values of `p`, the behaviour should gradually change from the behaviour at `p=0.5` to either `p=0` or `p=1` depending on the value of `p`.

   Test your implementation on ‘`brightness_contrast.jpg`’.

   Hint: In `contrastAdjust`, when the `p` is greater than `0.5`, the intensity mapping will be similar to Figure 1 (slope of the red line in the middle > 1). Similarly, find a curve (slope < 1) that can handle `p<0.5`.

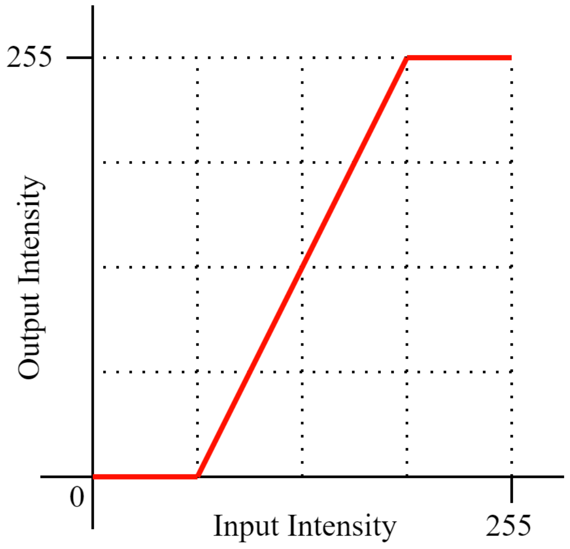

Figure 1: Curve to boost contrast

(20 Marks)