E9 241 Digital Image Processing
Assignment 02

Due Date: September 26, 2022 - 11:59 pm **Total Marks:** 70 + 15

Instructions:

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

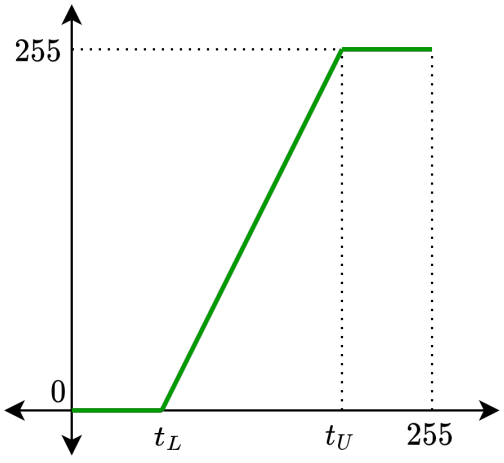
- Your function should take the specified parameters as inputs and output the specified results.
- Also provide the wrapper/demo code to run your functions. Your code should be self-contained, i.e., one should be able to run your code as is without any modifications.
- For python, if you use any libraries 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 and inferences. Include answers to subjective questions, if any.
- Put all your files into a single zip file and submit the zip file. Name the zip file with your name.

---

1. **Contrast Enhancement:** Apply the below contrast enhancement algorithms. Plot the input and output histograms along with them. For every image, mention which of the contrast enhancement algorithms succeeded and which of them failed. Also, explain why the algorithm succeeded or failed. You can get hints from the histogram plots. Make sure that the `matplotlib pyplot` function is not implicitly stretching the contrast of images while showing the output. Make use of the `vmin`, `vmax` arguments in the `pyplot` function.

   (a) Perform **Full Scale Contrast Stretching** (FSCS) on `ECE.png` and `IIScMain.png`. Does it work well on both images? Why not?

      i. **Saturated contrast stretching (Bonus - 10M)** You have to perform the following input-output mapping on the image using parameters t_L (lower threshold) and t_U (upper threshold).



      Determine t_L and t_U from the input image by applying some heuristics on the CDF of its normalized histogram. Perform the mapping on `IIScMain.png`.

   (b) Perform **Histogram Equalization** on `lion.png`, `Hazy.png` and `StoneFace.png`, and comment on them.

(c) **Contrast Limited Adaptive Histogram Equalization** (CLAHE): Divide the image into 8×8 non-overlapping blocks (for e.g., for an 80×80 image, there are 64 blocks of size 10×10. There will always be 64 blocks only). Perform Histogram equalization on each block. You will need to set a clip limit for CLAHE. Also, repeat the same by dividing the image into blocks with 25% overlap. Visually analyze the effect of changing the clip limit on CLAHE without overlap and CLAHE with overlap on images `lion.png` and `StoneFace.png`, and comment on the results.

Contrast enhancement functions:
Input: Grayscale image, parameters of the function (optional)
Output: Enhanced Grayscale image

(5+10(Bonus)+10+20 Marks)

2. **Image Upsampling with interpolation:** Downsample `flowers.png` by a factor of 3 and upsample them back to the original size using nearest neighbor interpolation and bilinear interpolation. Visually analyze the effectiveness of the interpolation methods by comparing upsampled images. Use a zoomed-in patch of the image for a better visual comparison.

Interpolated upsample function:
Input: Grayscale image, downsample factor, interpolation type
Output: Image that is downsampled and upsampled back to the original size.

(a) **Error maps (Bonus - 5M)** It will be easier to understand where the interpolation method failed using error maps. Create an error map that computes the point-wise absolute error between the input image and the upsampled image. In order to visualize the error map, apply contrast stretching algorithms to map them to a visible range. Compare the error maps for both interpolation methods and draw inferences from the results.

Error map function:
Input: Input image, upsampled image
Output: Contrast enhanced error map

(20+5(Bonus) Marks)

3. **Spatial Domain Filtering:**

(a) Use high boost filtering to sharpen the blurred image `blur.png`. Choose the scaling constant for the high pass component by qualitatively comparing the result with the clean image `earth.png`.

(b) What happens if you apply high boost filtering on `noisy.png` using the same scaling factor? What is the reason for this behavior?

High boost filter function:
Input: Image, mask size, scaling constant
Output: Sharpened image

(10+5 Marks)