CS 470: Assignment 01

Programming Assignments (90%)

Your program MUST be written in **Python**.

Your code file should be named **A01.py**. The goal of this assignment is to perform various intensity transformations. You will also create a Gradio interface of your own design to demonstrate the functionality of your code.

Restrictions

You CANNOT use the following functions:

- **cv2.equalizeHist**
- **cv2.calcHist**
- **cv2.LUT**
- **Anything from the cv2.intensity_transform module**
- **np.histogram**

You MAY use np.bincount, np.cumsum, np.sum, np.interp, np.arange, np.clip, np.log, np.round and other np functionality. ***If you are not sure whether a library function is permitted, ASK ME!***

For efficiency reasons, while and for loops must be avoided. Everything can be implemented by taking advantage of numpy's indexing and broadcasting functionality.

A01.py

All intensity transformation creation functions should return a numpy array of length 256 with dtype "uint8". **This is a lookup table (LUT).** ***You must also ensure that floating values are rounded, clipped to [0, 255], and converted to unsigned bytes.***

All input images are grayscale images of shape (height, width) and with dtype "uint8". Output images are expected to be the same.

You are welcome and in fact encouraged to write helper functions for this assignment, BUT the below functions should be written as specified:

- **def get_log_transform(max_r)**
  - **max_r** = The maximum input value that will map to 255
  - Compute the corresponding c value: $c = \frac{255}{\log(1+max\_r)}$
  - Return the appropriate LUT for the log transform:

$$s = c \log(1 + r)$$

- **def get_gamma_transform(gamma)**
  - **gamma** = The gamma exponent
  - Return the appropriate LUT for the gamma/power-law transform:

$$s = 255 * \left(\frac{r}{255}\right)^{gamma}$$

- **def get_hist_equalize_transform(image, do_stretching)**
  - **image** = Grayscale unsigned byte image
  - **do_stretching** = If True, do histogram stretching
  - Return the appropriate LUT for histogram equalization.
- **def get_piecewise_linear_transform(points)**
  - **points** = A ***2D Python list*** of (r, s) points that make up a piecewise linear transformation.
  - *HINT*: Sort the points by r, take advantage of zip to get lists of r and s values, and leverage np.interp to get the intermediate points.
  - Return the appropriate LUT for the specified piecewise linear transformation.
- **def apply_intensity_transform(image, int_transform)**
  - **image** = Grayscale unsigned byte image
  - **int_transform** = unsigned byte LUT (as a numpy array)
  - Return the transformed image (should be the same shape and type as original image).
- **def main():**
  - When running A01, your program should start a Gradio interface (that can be accessed through a browser; see the terminal for the address).
  - You are largely free to design this Gradio interface as you see fit with the following requirements:
    - You must allow the user to load an image.
    - You must allow the user to choose between histogram equalization (with and without stretching), gamma (with adjustable exponent), and log (with adjustable max_r value).
    - You must display the transformed image.
    - The interface should work correctly and as expected.
    - The interface should be reasonably organized.
  - BONUS CREDIT:
    - Display the input and output histograms.
    - Display the transformation function.
    - Add piecewise linear transformation.
    - Live updates of the histograms, transformation function, and/or output image.

Testing Screenshot (5%)

I have provided several files for testing:

- **Test_A01.py** – the test program
- **General_Testing.py** – basic testing functionality
- **assign01/**
    - **images/**
        - Sample input images
    - **output/**
        - Corresponding output images from various transformations

Once you have merged from my repo into yours, these files/folders should reside in your main project directory. Your source file (A01.py) should reside in the main project directory as well (NOT the assign01 folder).

You can either run the testing programs directly OR you can use the testing section of Visual Code.

You MUST run the tests and send a screenshot of the test results! Even if your program(s) do not pass all the tests, you MUST send this screenshot!

You may have to do "Command Palette" → "Python: Configure Tests" → unitttest → root directory → Test_*py

This screenshot should show clearly:

- The final result of the test run on the command line ("OK" for all passing, "FAILED (failures=N)" for some or all failing).
- OR
- The testing view in Visual Code (see image on right)

The screenshot should be copied to the screenshots/ folder.

WARNING: Passing the tests does NOT guarantee your assignment code is working 100% correctly! You are responsible for manually verifying whether your output visually matches what it ought to be.

The test programs also do NOT test the Gradio functionality.

README.md (5%)

Update the README.md of your repository with the following:

- Change "Your Name Here" to your actual name.
- Add a new entry under "Runnable Python Scripts" (similar to the one already there for BasicVision.py) for "A01.py". Very briefly describe what functionality is available and what happens if you run the program. I am only looking for a few sentences.

Grading

Your OVERALL assignment grade is weighted as follows:

- 5% - Testing results screenshot
- 5% - README update
- 90% - Programming assignments

I reserve the right to take points off for not meeting the specifications in this assignment description.

The following specific penalties (applied to the WHOLE assignment) can be expected:

Problem	**Maximum Penalty**
Code that is not syntactically correct	60
Usage of forbidden function(s)	60
Incorrectly implemented LUT generation functions	60
Usage of for/while loops	10
Incorrectly implemented apply_intensity_transform() function	10
Bad Gradio interface	5
Technically correct code, but poor code design	5
Missing assignment Git branch	5
Missing testing screenshot	5
README update missing/incorrect	5

Bonus credit of up to 5% for the whole assignment is available for exceptional Gradio interfaces. **Bonus credit will only be awarded if other functionality is correctly implemented.**

In general, these are things that will be penalized:

- **Code that is not syntactically correct (up to 60 points off!)**
- Sloppy or poor coding style
- Bad coding design principles
- Code that crashes, does not run, or takes a VERY long time to complete
- Using code from ANY source other than the course materials
- Collaboration on code of ANY kind; this is an INDIVIDUAL PROJECT

- Sharing code with other people in this class or using code from this or any other related class
- Output that is incorrect
- Algorithms/implementations that are incorrect
- Submitting improper files
- Failing to submit ALL required files