CS 470: Assignment 01

Programming Assignments (95%)

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

Your code file should be named **A01.py**. The goal of this assignment is to perform histogram equalization.

You may NOT use OpenCV functionality for histogram equalization! However, you may use image loading, image saving, and grayscale conversion; you may also use image window functions for debugging. ***If you are not sure whether a library function is permitted, ASK ME!***

A01.py should contain the following functions:

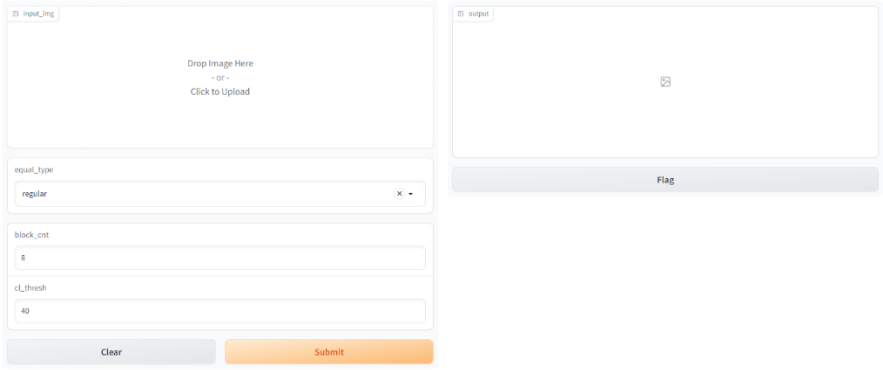
- **def create_unnormalized_hist(image)**
  - You can assume image is a grayscale image of shape (height, width) and uint8 dtype
  - Create a numpy array of type "float64" and shape (256,)
  - Given the provided image, return the UNNORMALIZED histogram
- **def normalize_hist(hist)**
  - Given an unnormalized histogram, use the sum of all elements to return the normalized histogram (you may use np.sum)
- **def create_cdf(nhist)**
  - Given a NORMALIZED histogram, compute and return the CDF (as a numpy array of shape (256,) and type "float64")
- **def get_hist_equalize_transform(image, do_stretching)**
  - Use create_unnormalized_hist to calculate the unnormalized histogram
  - Normalize your histogram
  - Make the CDF
  - If do_stretching is True:
    - Perform histogram stretching on the CDF
  - Create your transformation function by:
    - Multiplying the CDF by 255.0
    - Using the following to convert it to a 1D numpy array of uint8:
      - int_transform = cv2.convertScaleAbs(int_transform)[:,0]
  - Return your intensity transform
- **def do_histogram_equalize(image, do_stretching)**
  - COPY your image → output
  - Get your transformation function
  - For each pixel in the image
    - Get the value

        - Use your transformation to get the new value
        - Store it into the OUTPUT image
    - Return the output image

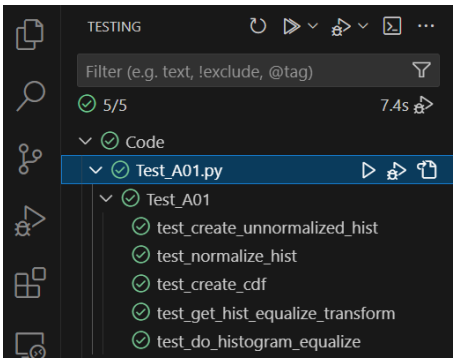
In addition to the functions, you will also copy and paste in the following to allow you to run your program with Gradio (be sure to "import gradio as gr" at the top):

```
def intensity_callback(input_img, do_stretching):
    input_img = cv2.cvtColor(input_img, cv2.COLOR_BGR2GRAY)
    output_img = do_histogram_equalize(input_img, do_stretching)
    return output_img

def main():
    demo = gr.Interface(fn=intensity_callback,
                        inputs=["image", "checkbox"],
                        outputs=["image"])
    demo.launch()

if __name__ == "__main__":
    main()
```

If you run your program, you should be able to open a web browser to http://127.0.0.1:7860 and see a nice GUI version:



Testing Screenshot (5%)



I have provided several files for testing:

- **Test_A01.py** – the test program
- **General_Testing.py** – basic testing functionality
- **assign01/**
    - **images/**
        - Input images for testing
    - **ground/**
        - Ground-truth data and images

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.

Grading

Your OVERALL assignment grade is weighted as follows:

- 5% - Testing results screenshot
- 95% - Programming assignments

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

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