CSC 481 – Intro. To Image Processing
2024-25 – Winter Term
Instructor: Kenny Davila Castellanos

Weekly Assignment 3: More Image Transformations

Objective

The main objective of this assignment is to help student get familiar with multiple image transformations such as histogram-based analysis, filtering, and bit plane splicing.

Data

To complete this assignment, please pick 3 images of your liking. You should use:

1) A picture from any wild animal species that you find interesting. This can be a picture taken by you or from the internet. Please clearly indicate the source on your report.
2) A picture from any pet animal species that you find interesting. This must be a picture taken by you or by one of your friends. This should be different from the first picture.
3) A random picture of your liking different from the first two. Ideally, look for an image that has interesting objects of different intensities.

You will have to run your code with each of these images, and then you must display and briefly discuss the results in your report.

Part 1 – Histogram Equalization

For each image, do the following main steps:

a) Read and display the image.
b) If not grayscale, make it grayscale.
c) Calculate and display the histogram of this image.
d) Enhance the contrast of the intensity image using histogram equalization and display both the uniform histogram and the newly enhanced intensity image.
e) In the **report**, explain why the two histograms (of the original image and of the enhanced image) are different.
f) Calculate the difference between the original and enhanced image. Display it.

In addition, (f) apply a local enhancement approach on this image and show your results. Before you start, consider how your image might call for a particular window size. For fun, you might want to try a few different window sizes. Make sure to **explain in your report** how did you decide what local enhancement and window size you used.

For **every input image** (you have 3 inputs, see **Data**), you must show: the original image and its histogram, the equalized image and its histogram, the difference image, and (CSC 481 only) a modified version of the image using a local enhancement.

Part 2 – Image Filtering

For students in the CSC 481, you will have to write a function that does linear image filtering for **grayscale images**. Do not use any of the prewritten functions for filtering. Your function should process the input image **pixel by pixel** and explicitly calculate the **sum of products** for each pixel in the image. You can use whatever simplifying assumptions you want for pixels on the border of the image (produce a smaller image, use zero-padding, etc.). Make sure that you **explicitly state** those assumptions in your **report**.

Then, you must perform filtering of the **grayscale version** of your image using both your custom function and a built-in filtering function with the following filters:

1) Prewitt filter.
2) Sobel filter.
3) Point filter.
4) Blurring filter.

Calculate a difference image between the outputs of your filtering function and the built-in function. Note that if you use padding in your custom function, you also need to add padding on the built-in function so that their outputs have matching image sizes.

For **every input image** (you have 3 inputs, see **Data**), you must show: the original image, and the two filtered images (your function and the built-in function) plus the difference image ([4 image filters] x [3 images per filter] = [12 results per input image]).

Part 3 – Bit Plane Splicing

Bit place splicing (https://en.wikipedia.org/wiki/Bit_plane) is a simple form of frequency analysis in which the frequencies are defined by the bits representing the intensity of the pixels. Write a program to perform bit-plane splicing on the grayscale version of the input image, such that you can generate a figure similar to the one shown in the Wikipedia article: your original (**grayscale**) image and each of the 8-bit planes in it. Each bit-position represents a different binary image. Some bit-level manipulation and additional math will be required to display the plane images as black and white images.

Then, “assemble” the original image by successively adding bit planes to the most significant bit plane. You will have 7 new images, which will be the combination of bit planes 7 and 6; 7 and 6 and 5; 7 and 6 and 5 and 4; … all bit planes, which should be the original image. Again, some math will be needed to make sure that your results are correct. State which bit plane you feel you could stop at and still get a good visual match with the original image.

For **every input image** (you have 3 inputs, see **Data**), you must show: the original image, the bit planes images (8 images), and the results of gradually assembling the image (7 images).

The Program

In this assignment, you will write a program that handles multiple operations. You can use any programming language of your preference, but Python is highly recommended. Your code should be properly organized and well-commented to clearly identify the correspondence between portions of the code and each part of this assignment. The code should compile and run on any properly configured programming environment without the need of making significant changes to it. In this sense, it is highly recommended to avoid the usage of absolute paths.

The report

You are asked to submit a written report showing the results for each part of the assignment. Use proper section headings and descriptions to clearly identify the results of each part. If I cannot easily find a match in the report for a required result, I might assume that this portion was not completed, and a score of 0 will be given by default.

For students working with Python, you are allowed to use Jupyter Notebooks. These allow you to combine code with the report in a single deliverable, which is perfectly acceptable for this course. Besides the original Jupyter notebook file, please also submit an export to PDF.

Delivery Instructions

You are given the freedom to use any programming language and IDE of your preference. You are also required to provide your own images, and to produce a written report. When done, **you should submit everything using a single Zip file**.

File names. The zip file that you submit should use "[Last Name(s)], [Given Name(s)].zip" as it appears in D2L. For example, "Kenny Davila Castellanos" (Davila Castellanos is two last names), would have to submit the homework with the name "Davila Castellanos, Kenny.zip". Another student named "Kenny Mauricio Davila" (Mauricio is a middle name), would have to submit the homework as "Davila, Kenny Mauricio.zip". Not following these instructions might lead to a penalty.

Policies

1. All general policies about Plagiarism and Cheating apply to this homework. If you plagiarize or receive code from other people, you will be caught and you will receive a score of 0, and a report of the academic integrity violation will be filed.
2. Please limit the usage of Chat-GPT and other code generators to ethical usage only. Submitting code that was directly generated by these tools is considered a form of plagiarism.
3. Do not post your solutions online and do not share them with anyone. It is your responsibility to safeguard your private data.
4. **Code that does not compile due to syntax and/or semantic errors will automatically receive a score of 0.** It is hard to assign partial credit when I cannot even run your code.
5. You can use any programming language of your choosing.
6. You must follow the delivery instructions.

7. Very late homework's without justification will receive a score of 0.
8. The homework description outlines very specific requirements. You are welcome to try other things and report your results. However, no amount of extra work can be used as a substitution of the actual requirements.
9. Do ask for help if anything is unclear, but do it in a timely manner (e.g., by e-mail, Discord or during the Office Hours).