Name: Ishtiaq Shahriar
Course: CS 583
Subject: Assignment 1

Image Processing

Abstract

The goal of this project is to get familiarized with image processing in python with numpy and gain experience implementing low-level image filtering techniques.

Introduction

In this project we develop a low pass filter to apply gaussian blurring on images. The skeleton code for the project was provided. We had to create the logic for the following functions:

- Load_image
- Create_gaussian_kernel
- Convolution_pixel
- Convolution
- Split
- Merge

The program was designed to function based on two user inputs, the size of the kernel to be used and the standard deviation, sigma.

Implementation

Kernel
The dimensionality of the kernel to be designed had to be odd and the resultant values of the kernel matrix should have a sum total of 1. The low pass filtration formula shown below in figure 1, was used to implement the kernel. The sigma as mentioned in the previous section was provided by the user. The developed kernel was then passed into the convolution method.

$$G(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

Fig 1: Low pass filter for gaussian blurring

Image

The image was loaded into the program using the Pillow library. It was then converted into a numpy array and passed on to the split method. The split method separates the image into 3 channels red, green and blue. Each of these channels (numpy arrays) were then passed into the convolution method.

Convolution

The convolution method takes a single channel and the kernel as inputs. A nested for loop is used to iterate through each location in the channel matrix. The convolution_pixel method is called on each location of the channel matrix. The convolution_pixel takes the single channel, the kernel and the channel matrix location as inputs.

The convolution_pixel method is designed such that, if the location inputted is on the edge of the channel, no kernel convolution takes places and simply returns the original value of that location. If the location doesn't meet the above criteria then it passed on to a convolution algorithm. Keeping the provided location as the center, the kernel is convoluted with the inputted channel. The convolution_pixel method returns the result of the convolution back to the convolution method.

The convolution method assigns the output from the convolution_pixel, to the location of the channel matrix initially inputed into the convolution_pixel.

Merge

After each channel passes through the convolution method, they are inputted in the merge method. The merge method combines the channels and outputs a RGB image numpy array. This array is then passed into an ImageIO method to be saved as a jpeg image.

Results

Following are the two images used for testing purposes.

Input Images

Figure 2: Input image one

Figure 3: Input image two

Output Images

The input images were used to perform three tests with different input variables.

Figure 4: Output images with inputs variables k = 3, sigma = 1

Figure 5: Output images with inputs variables k = 5, sigma = 2

Figure 6: Output images with inputs variables k = 7, sigma = 3

Figure 7: Output images with inputs variables k = 3, sigma = 1

Figure 8: Output images with inputs variables k = 5, sigma = 2

Figure 9: Output images with inputs variables k = 7, sigma = 3

Conclusion

This project was a great starting off point for someone really interested in computer vision. As I completed the project, I got refresher for a lot of concepts I learned during my time as an Undergraduate student.
This was a great learning experience and given me the confidence to take on more interesting projects in the future.
Below I have provided the test results for my program.

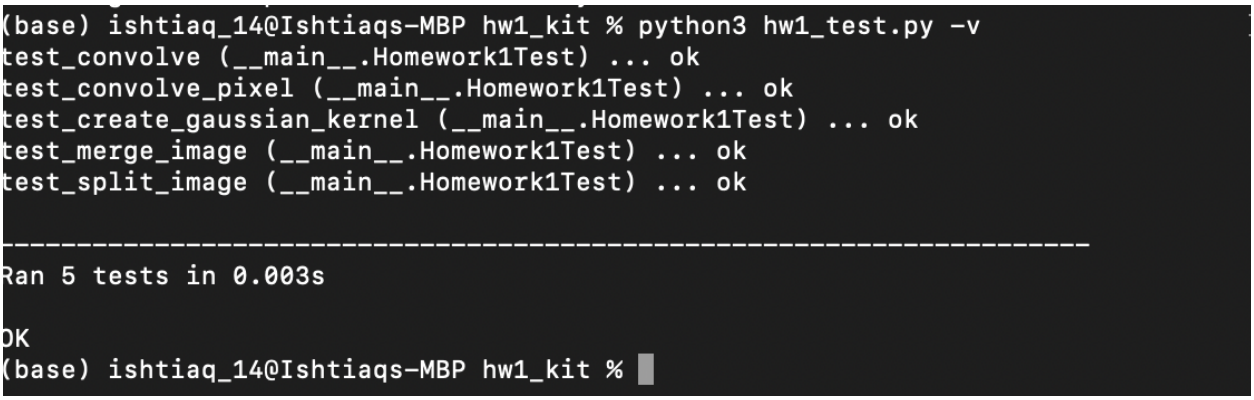

Figure 10: Test results for Hw_1