cs512 Assignment 1: Report

Amit Nikam
Department of Computer Science
Illinois Institute of Technology

October 8,2020.

Abstract

In the programming assignment 1, I have implemented a program that performs image manipulations using python. Concepts related to computer vision like edge detection, noise and filtering, color formation and simple image transformations were put to test. To achieve the outcome, libraries like OpenCV, NumPy and MatplotLib were used. In this report I covers all the problems I faced while implementing the program, my solutions to these problems, complete implementation and the outcome.

1. Problem Statement

As a part of the problem, image manipulations were to be performed on the image. Some of these tasks involving converting the images to grayscale, smoothing the images, scaling them and taking their derivatives.

The first challenge while implementing the program is converting the image to grayscale without using OpenCV. OpenCV does this task simply in one line of code, but to manually do this process the image array needed to be traversed, computed open and normalized, so its definitely not as simple.

Next problem is implementing smoothing using track-bars. The challenge here is to implement an algorithm which will take the track bar input and the image, and would create a kernel, a padded image if needed and then perform the operations of smoothing. For this problem an algorithm needs to be implemented from scratch.

This introduced another problem, implementing such a complicated task in base python would slow down the program, so a differ, faster implementation is required for this particular problem.

One other issue is that there is no way to down-sample an image in OpenCV without interpolating it. For this reason I have to implement a way to down-sample the images from scratch.

Finally, one of the most challenging task is to get the gradient vectors and plot it along the image. This is particularly different from other problems because OpenCV will not plot such figures so a different method has to be used. All in all most of the other problems are where implementation from scratch is needed, rest all problems will be solved using OpenCV or NumPy.

2. Proposed Solution

To convert the image to grayscale without using library, I will be traversing and the image pixel by pixel and performing a computation that would convert each pixel to grayscale. The concept behind this computation is to take the weighted sum of the values of pixels in channels RGB and then normalize these values.

To implement smoothing of the image using a track bar to control the intensity, I will be implemented an algorithm that does averaging filter[1] of the image. In this i would be taking the size of the kernel from the user and the image, and perform smoothing. A kernel would be created from the size the user specifies and then, a padded image would be created upon which convolution with the kernel would be performed to get the smooth image. This is a complicated task as a lot of computation will be done, so to speed up the implementation cython[2] would be used.

To down-sample the image without smoothing by a factor of 2, the image would scanned pixel by pixel and every seconds row and column pixel would be eliminated. In this way the down-scaled image without smoothing will be achieved.

Finally to plot gradient vectors, matplotlib would be used. Matplotlib has a quiver plot[3] which will work perfectly to plot the vectors. The challenge is to get variable grids for vectors through track bar, but it can be achieved as matplotlib will be a different window. To achieve other solutions, OpenCV or NumPy libraries were used.

3. Implementation Details

The following files are included in src folder:
- Main.py
- self_func.pyx
- self_func.c
- self_func.so
- setup.py

Before running the main program, make sure to build the cython dependencies. To build the cython modules, simply execute the following command in the terminal in src direct:

```
$ python3 setup.py build_ext --inplace
```

After the cython files have been built, make sure that .so extension module(might be different for windows/mac) is named ‘self_func’.

Now the program is ready to run.

To use an image ‘filename’ located at ‘loc’ to process in the program, simply run the following command in the terminal:

```
$ python3 main.py --img loc/filename
```

Passing the image name as argument during initialization is required.

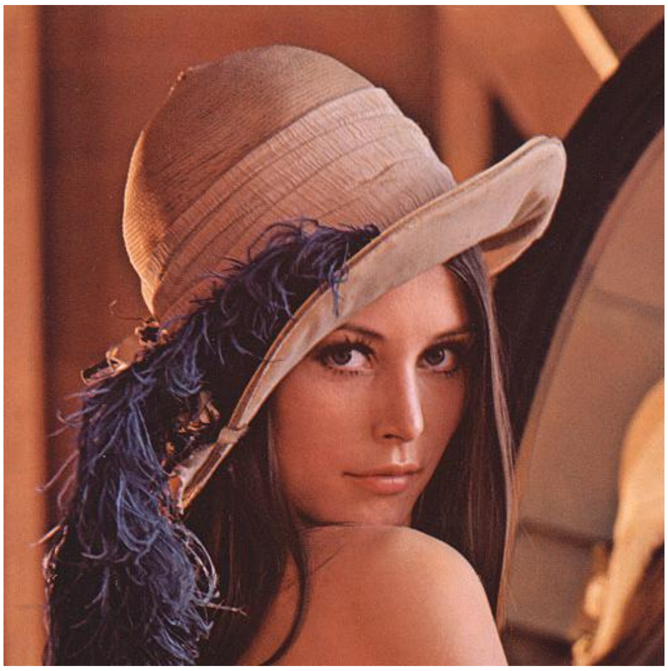
To use camera input to process, simply run the main program. It will automatically try to get camera response from front camera (0).

```
$ python3 main.py
```

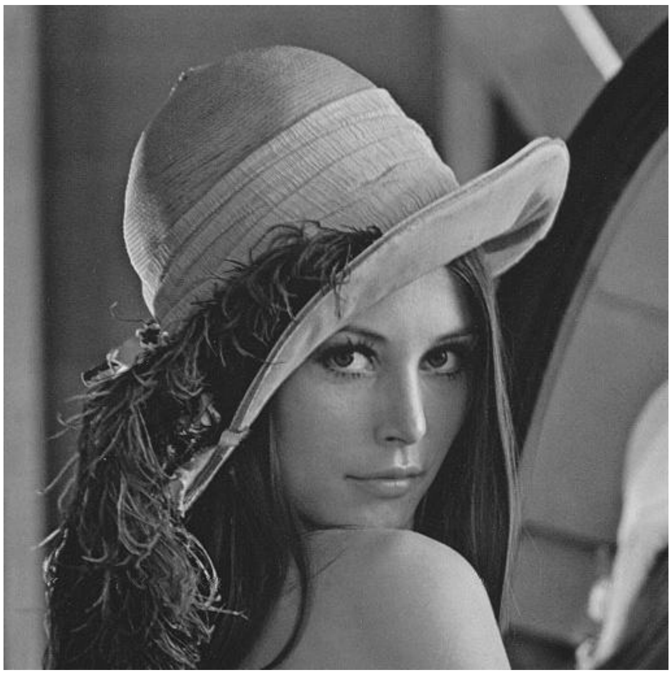
4. Results and Discussion

Image Conversion to grayscale:

Original:

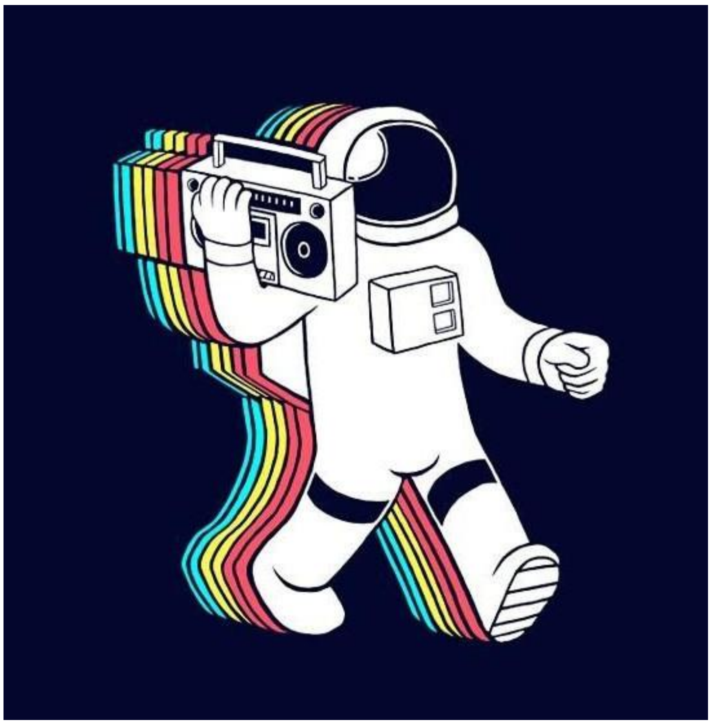


Result:

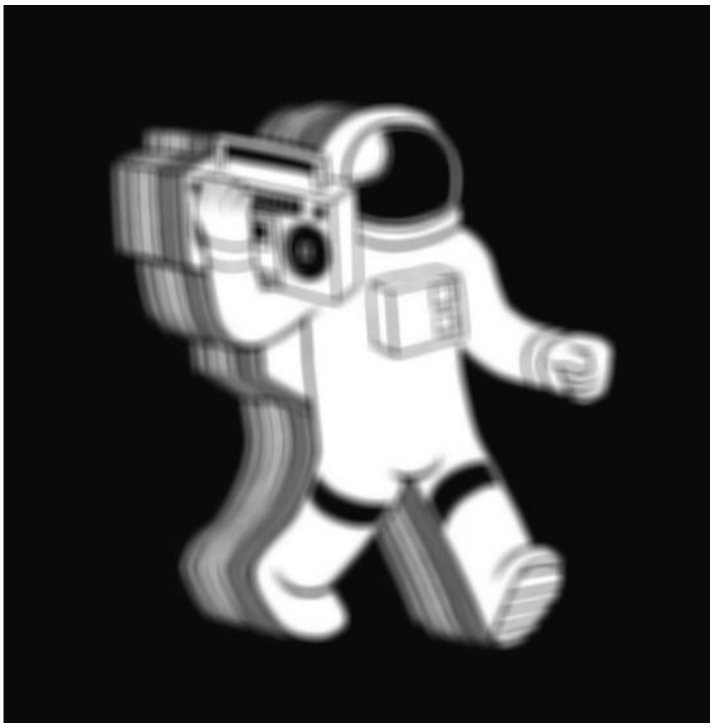


Smoothing Image

Original

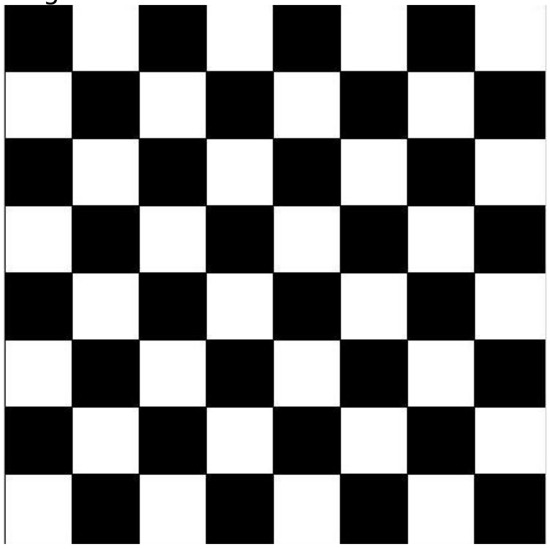


Result with kernel size 11:

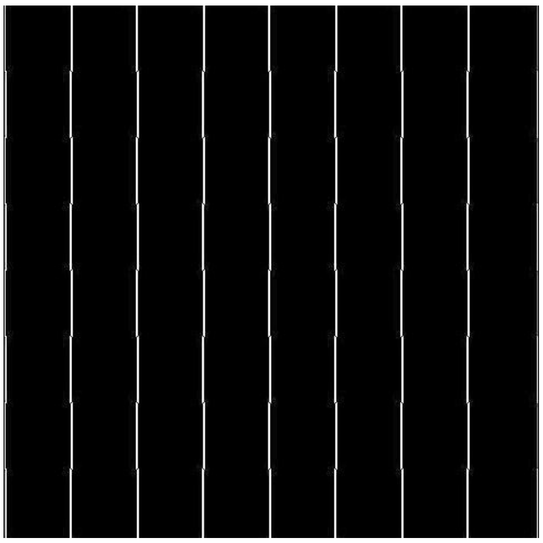


Derivatives of the image

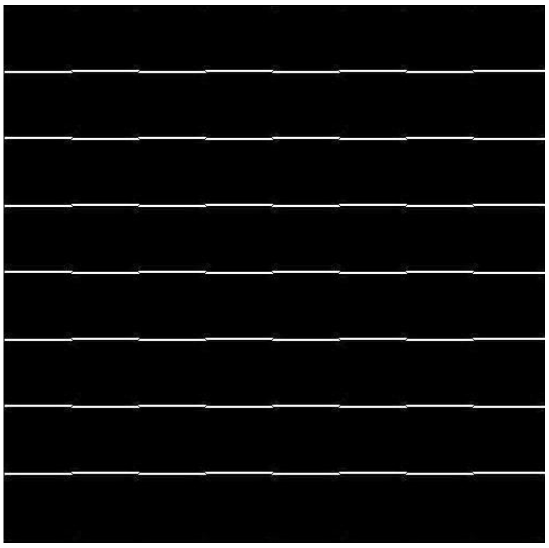
Original:



X Derivative:

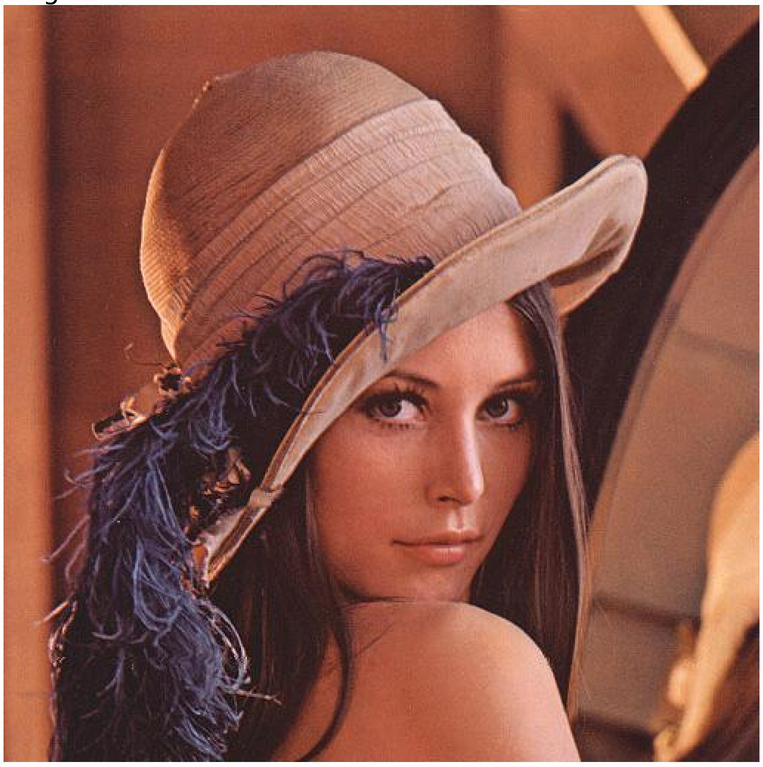


Y derivative:

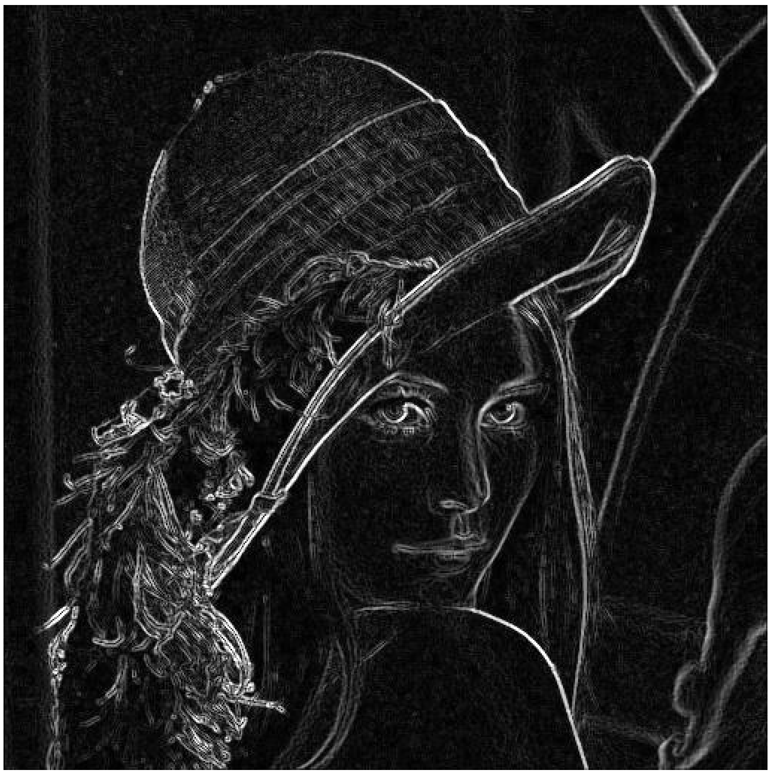


Magnitude of gradient normalized:

Original:



Magnitude of gradient:



5. References

[1]
https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_imgproc/py_filtering/py_filtering.html
[2] https://cython.org/
[3]
https://matplotlib.org/3.1.1/gallery/images_contours_and_fields/quiver_simple_demo.html#sphx-glr-gallery-images-contours-and-fields-quiver-simple-demo-py