The report of mini-project N2 program

by Ulvi Bajarani

SID 20539914

All retrieved results and processings are saved in two different log files created by the program (if the log files don’t exist, the program creates it. Otherwise, it appends): **logFile.txt** and **ResultslogFile.txt**

The program processes all images that must be in the **/imagesToProcess** folder located in the same folder with the **/python** folder. In other words, **/imagesToProcess** folder must be one level up than the code file. The .py file could be found in **/python** folder. The image processing results are saved in the **/resultsOfProcessing** folder. Separate folders for separate processes are created by the program. The **/python** folder contains the code files.

In the first stages, the program adds the noises. The first noise is Gaussian with mean=32 and variance=16. The second noise is Salt and Pepper with probability 0.1 and 50% Salt and 50% Pepper ratio.

Then, the program consecutively executes unweighted averaging, weighed averaging, Gaussian averaging (all three ones for the images with Gaussian noise), median averaging 3x3, median averaging 5x5 (all two ones for the images with Salt and Pepper noise), Sobel edge detection (for the images with no noise). All operations are done by OpenCV and then immediately by Formula. All operations are done in the grayscale images.

The time measures are calculated by the time.process\_time() function (**start\_time** and **stop\_time** variables), which returns the sum of the user and system CPU time of the running process. The **start\_time** and **stop\_time** variables exist in all processing functions. Below you can see the execution results:

|  |  |  |
| --- | --- | --- |
| The execution times for 25 images | | |
|  | By OpenCV approach | By Formula approach |
| Unweighted averaging | 2.03125 seconds | 2688.765625 seconds |
| Weighted averaging | 1.796875 seconds | 2525.65625 seconds |
| Gaussian averaging with sigma=16 | 1.671875 seconds | 3389.421875 seconds |
| Median filtering with 3 to 3 kernel | 1.90625 seconds | 543.796875 seconds |
| Median filtering with 5 to 5 kernel | 2.109375 seconds | 1281.90625 seconds |
| 3x3 Sobel filtering for 25 images | 86.8125 seconds | 3653.34375 seconds |

The differences between OpenCV and Formula approaches:

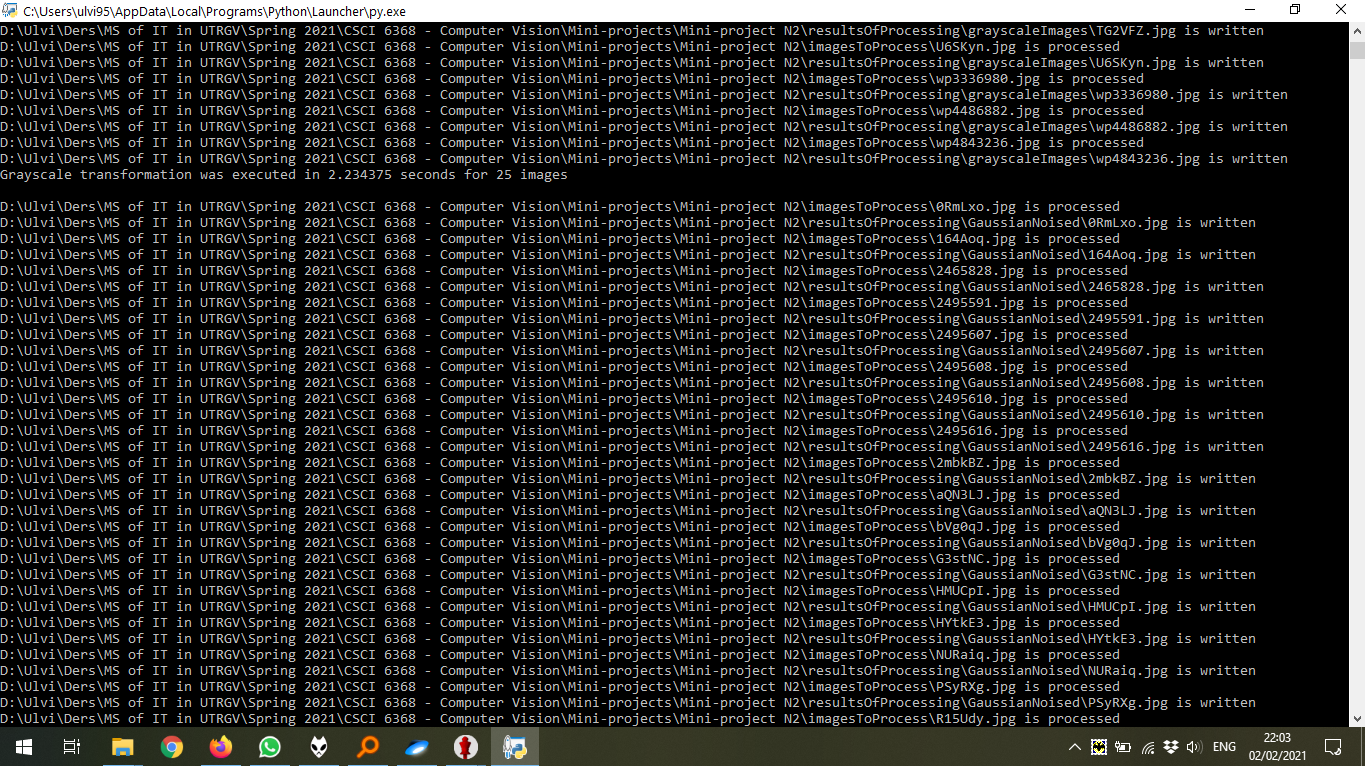
1) The speed of OpenCV approaches are dramatically higher, might reach to 1000(!!) times.

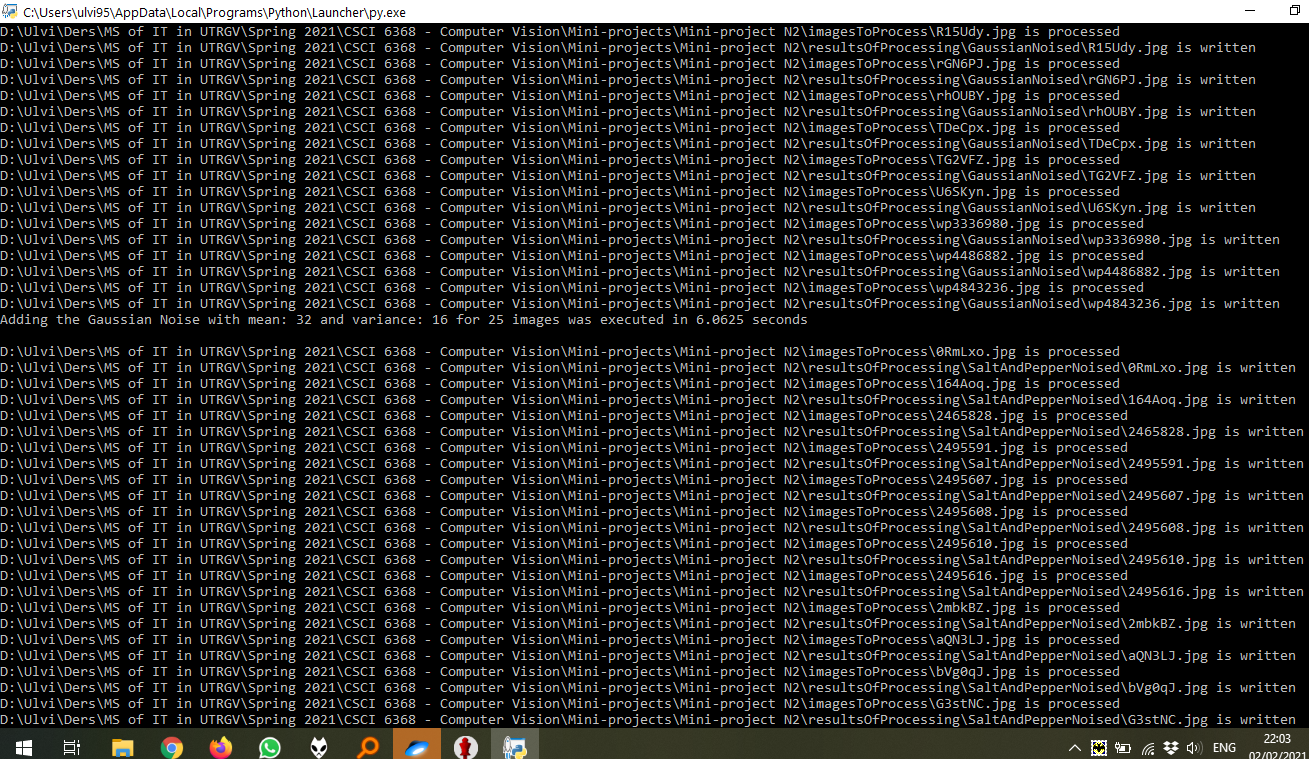
2) The differences in the borders. While OpenCV might implement some approach to handle them, in the formula based approach, the program copies them from the original image

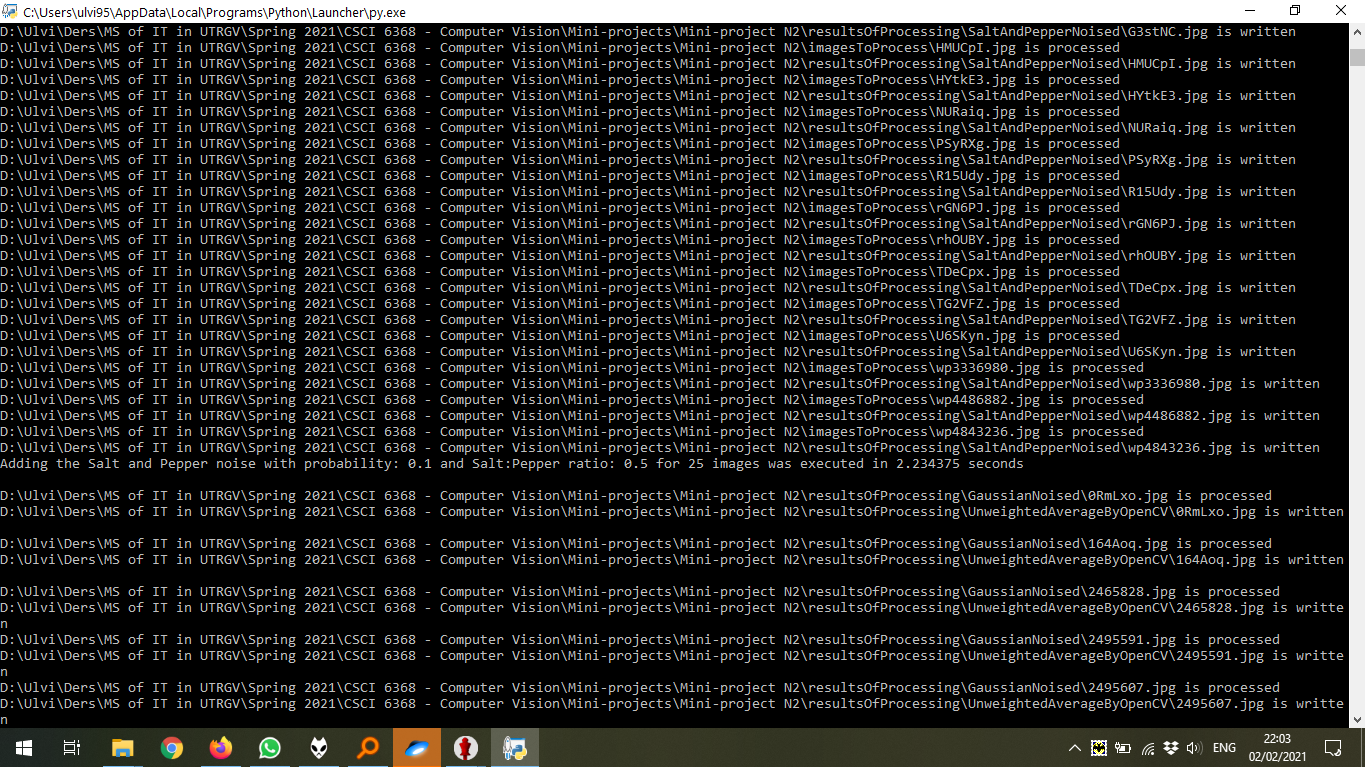
3) The non-border results in the Weighted and Gaussian averaging are slightly different (although it might not be distinguished with normal eyes). For Gaussian averaging, the reason might be the how floating values are handles. For the weighted averaging, the initial matrix implementation might be the reason (np.float32 in the OpenCV approach)

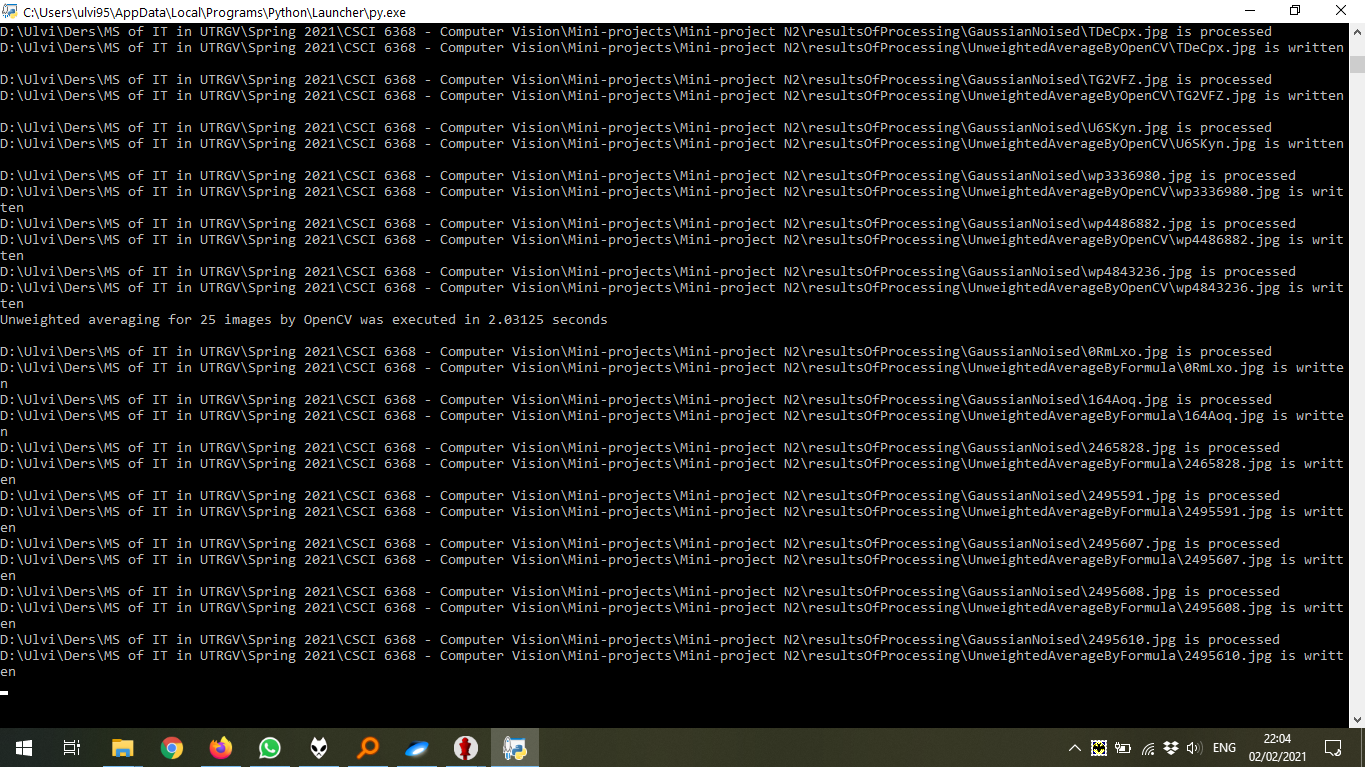
4) The formula approach allows to have specific size filters (for example, 3x5), while OpenCV approach is not such flexible.

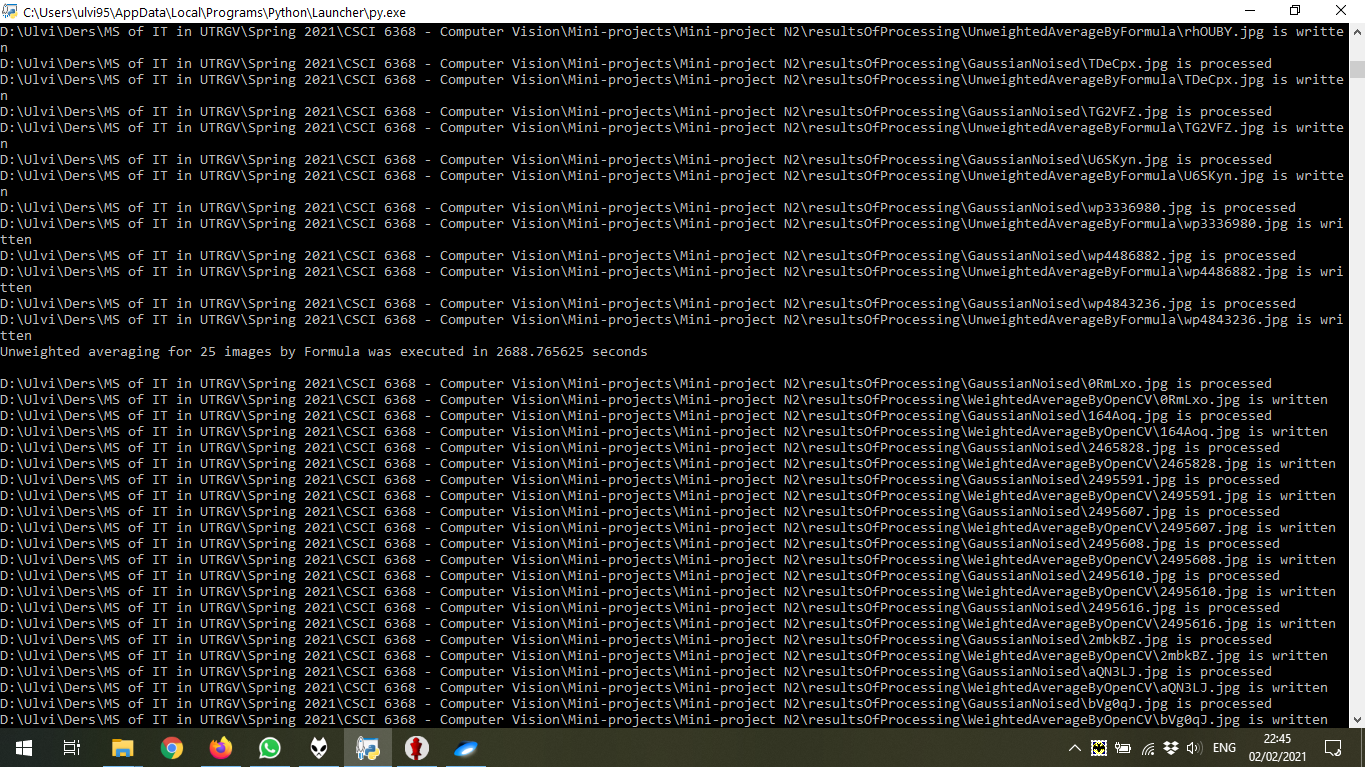
The program describes the actual state of processing. In the screenshots below, it can be seen that what processes ended and what are in the progress (by the name of the folders):

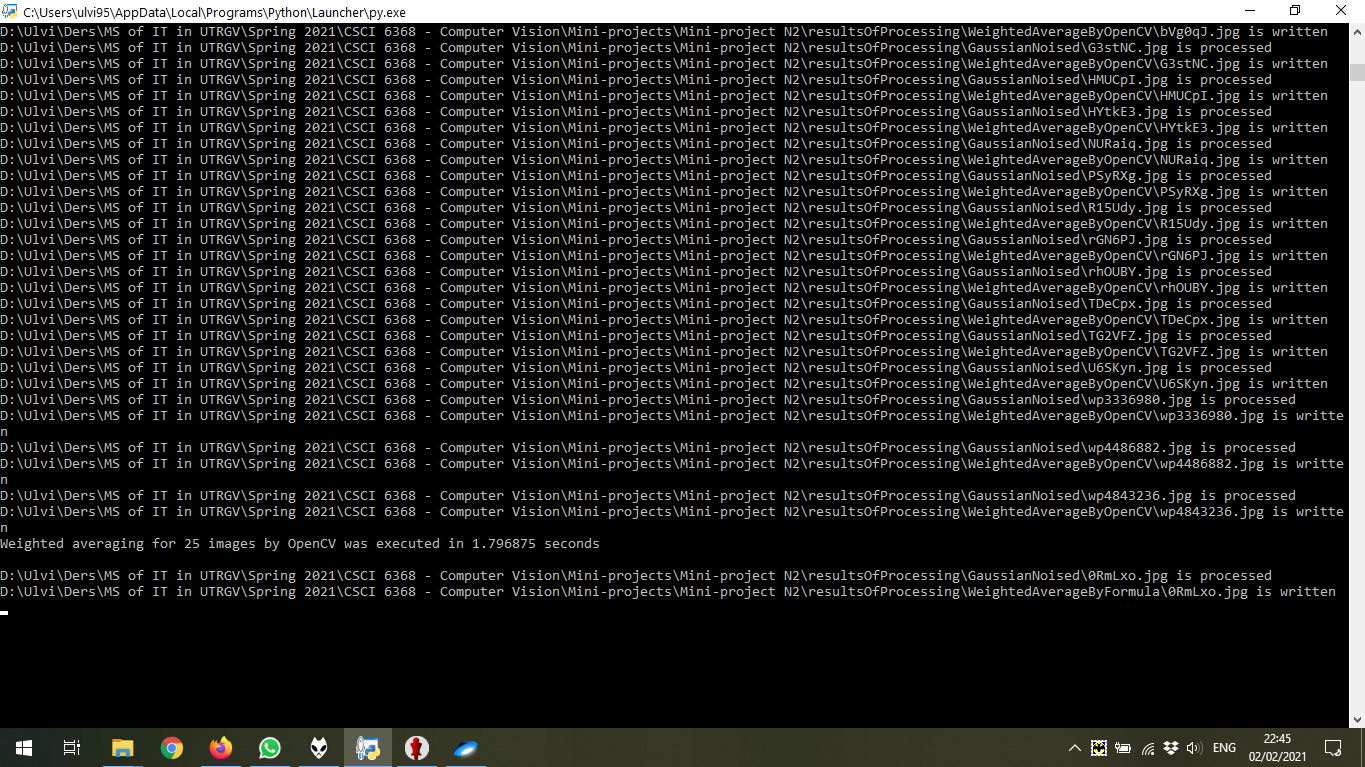












Below the thumbnails of results of processing for 1 image (out of 25) chosen randomly. As mentioned above, all results could be found in the **/resultsOfProcessing** folder.

|  |  |  |
| --- | --- | --- |
| Convolution Operation Comparison on the grayscale sample image | | |
| Sample Image (Gaussian Noised) | Unweighted Average by OpenCV | Unweighted Average by Formula |
|  |  |  |
| Sample Image (Gaussian Noised) | Weighted Average by OpenCV | Weighted Average by Formula |
|  |  |  |
| Sample Image (Gaussian Noised) | Gaussian Average by OpenCV | Gaussian Average by Formula |
|  |  |  |

|  |  |  |
| --- | --- | --- |
| Convolution Operation Comparison on the grayscale version of sample image | | |
| Sample Image (Salt and Pepper Noised) | Median Filtering 3x3 by OpenCV | Median Filtering 3x3 by Formula |
|  |  |  |
| Sample Image (Salt and Pepper Noised) | Median Filtering 5x5 by OpenCV | Median Filtering 5x5 by Formula |
|  |  |  |
| Sample Image (No Noise) | 3x3 Sobel Filtering by OpenCV | 3x3 Sobel Filtering by Formula |
|  |  |  |