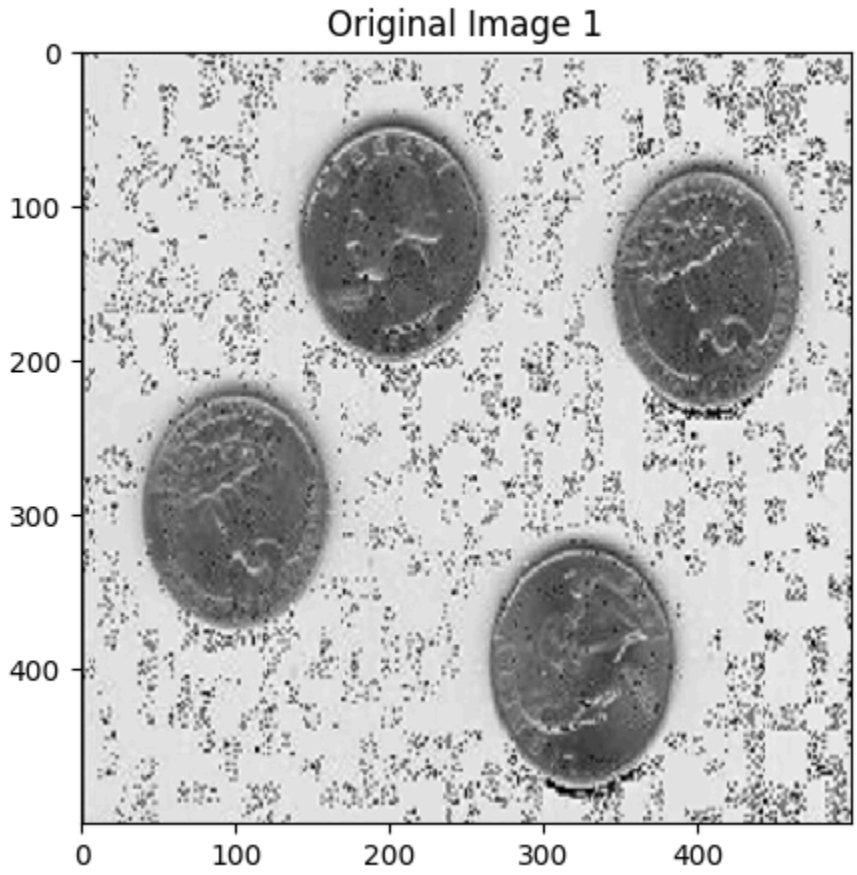
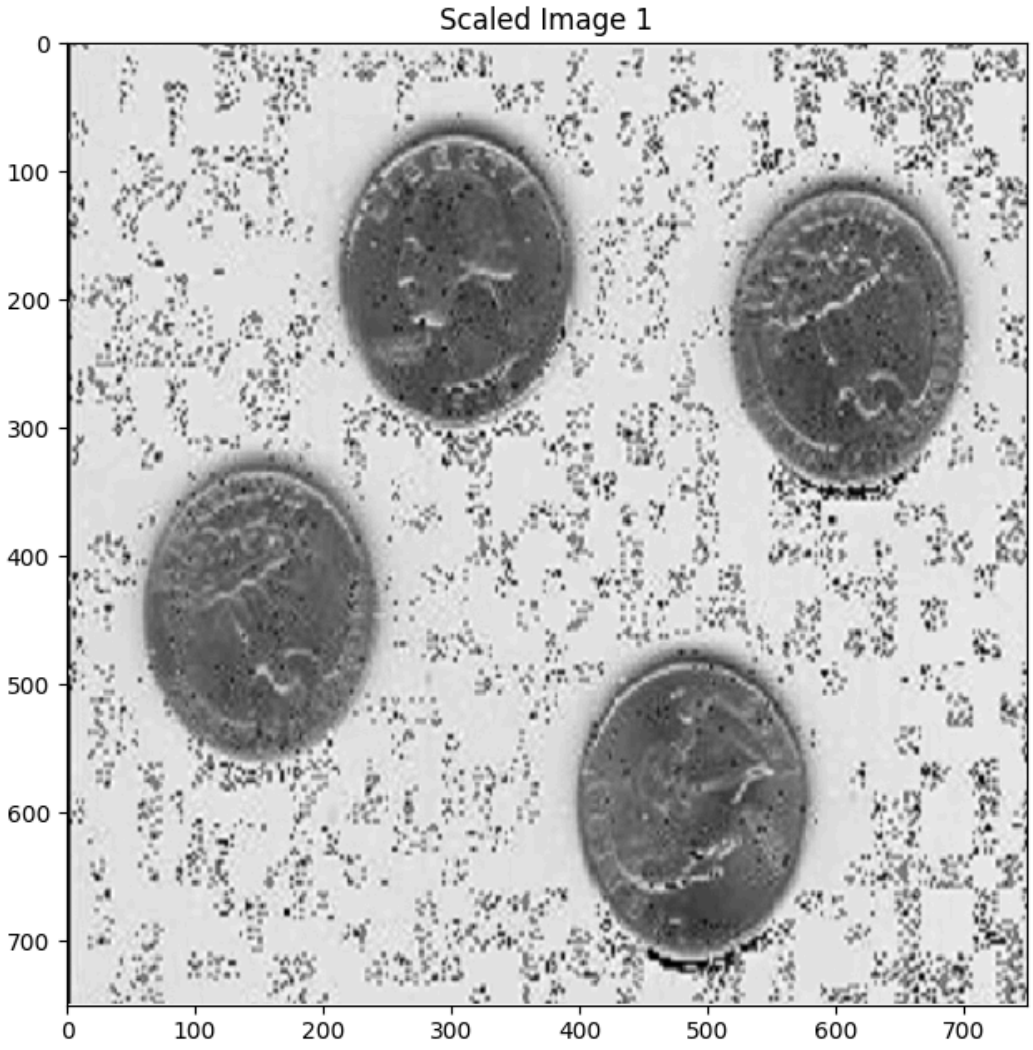
Ashkan Zarkhah
610399196
You can find all images and plots in the “plots” folder, and all the programming codes, in the “code” folder.
Report:

- **Geometrical spatial operations:** First, we started by loading Image 1 and visualizing it.



Then we started scaling the original image into a new one with 1.5 times width and height. To do so we used the resize function of the cv2 library

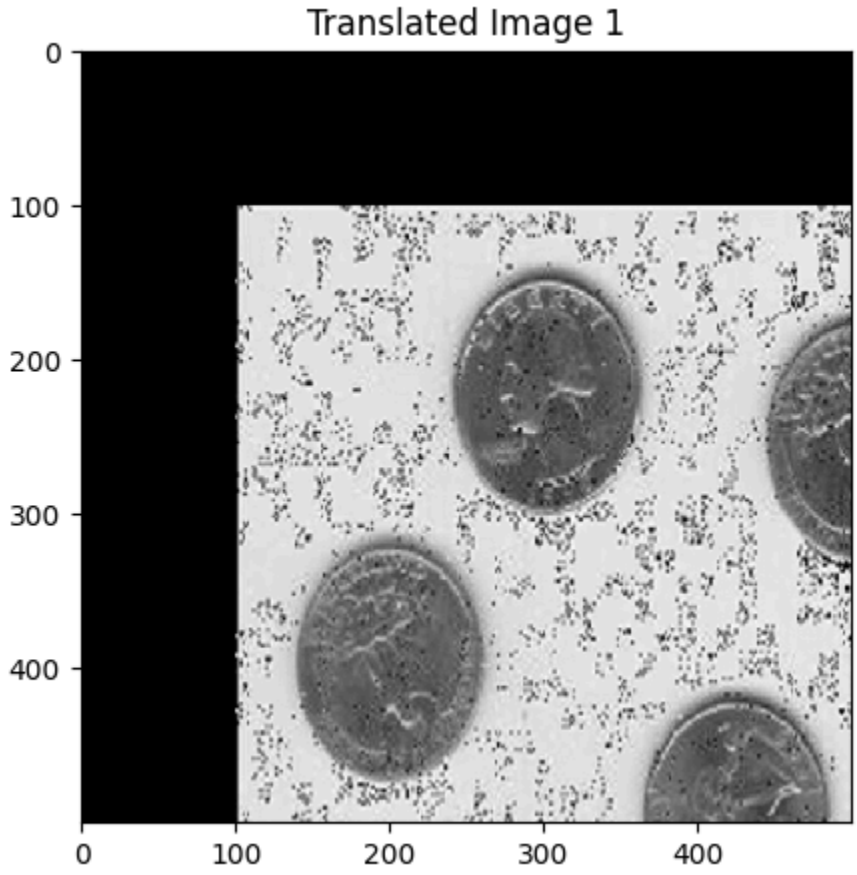
and scaled the image using the linear function. The achieved Image is:



After that, we Implemented the translation function. To do so, we used the warpAffine function of the cv2 library. Because there were no parameters to follow, we shifted our image 100 pixels left and 100 pixels down, so our translation matrix was:

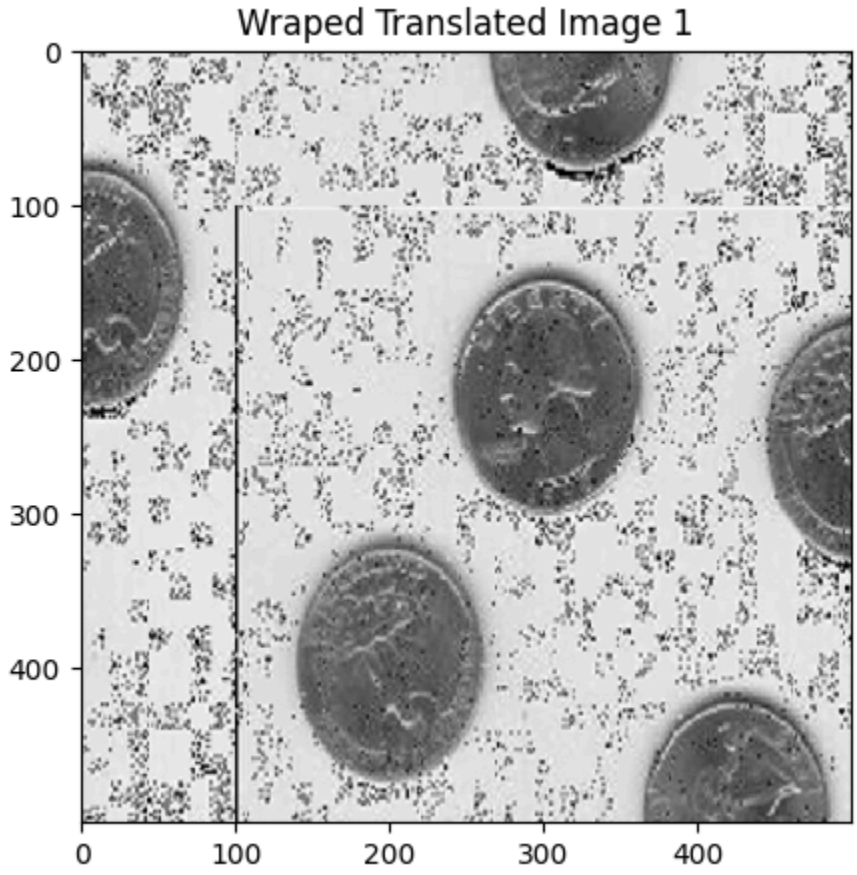
$$\begin{bmatrix} 1 & 0 & 100 \\ 0 & 1 & 100 \end{bmatrix}$$

And the achieved Image is:



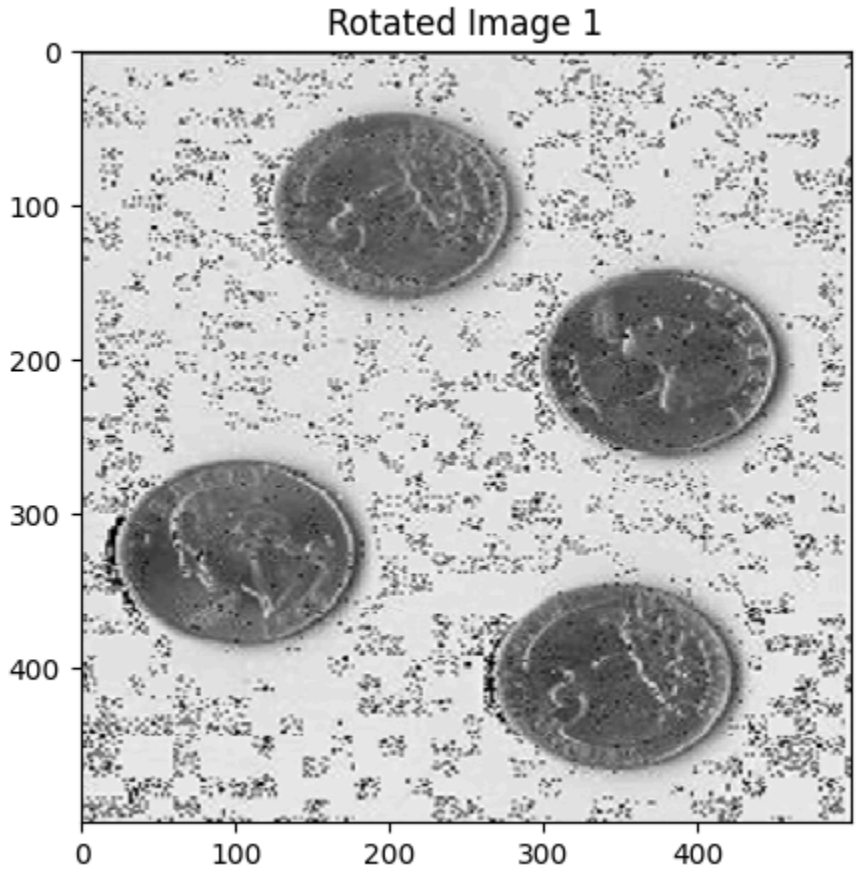
But then we realized we were losing information by just translating our image. So we decided to wrap the lost information around the image and

the resulting image is:

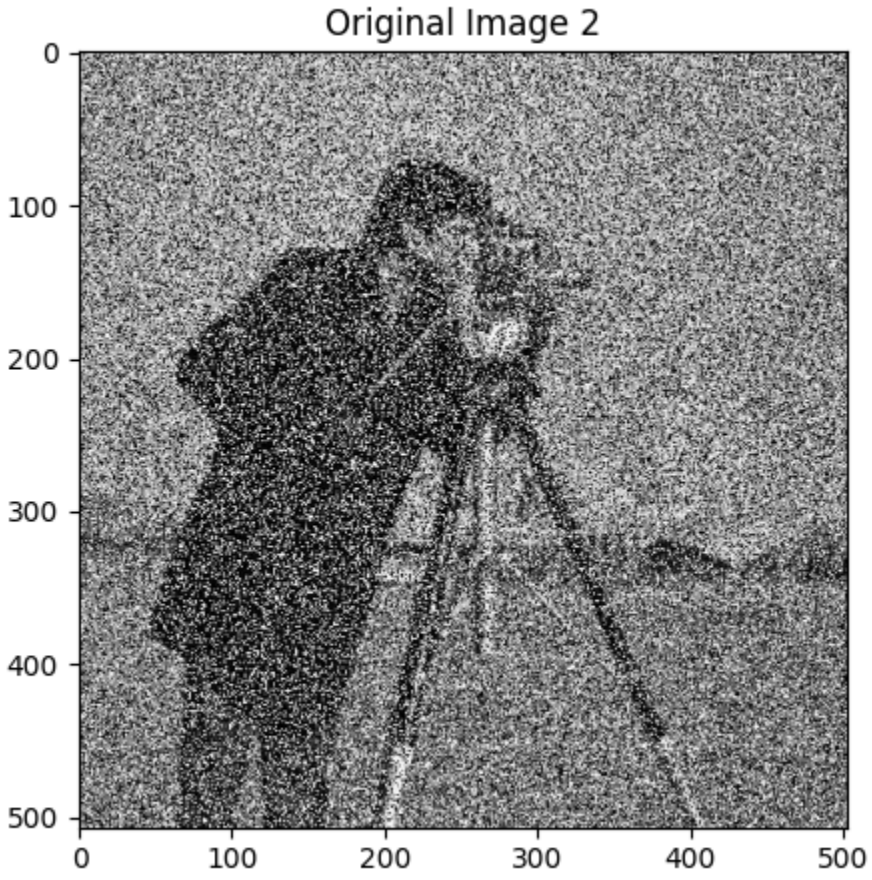


At last, we implemented the rotation of the original image. To do so, we used the rotate function of the cv2 library and we decided to rotate our

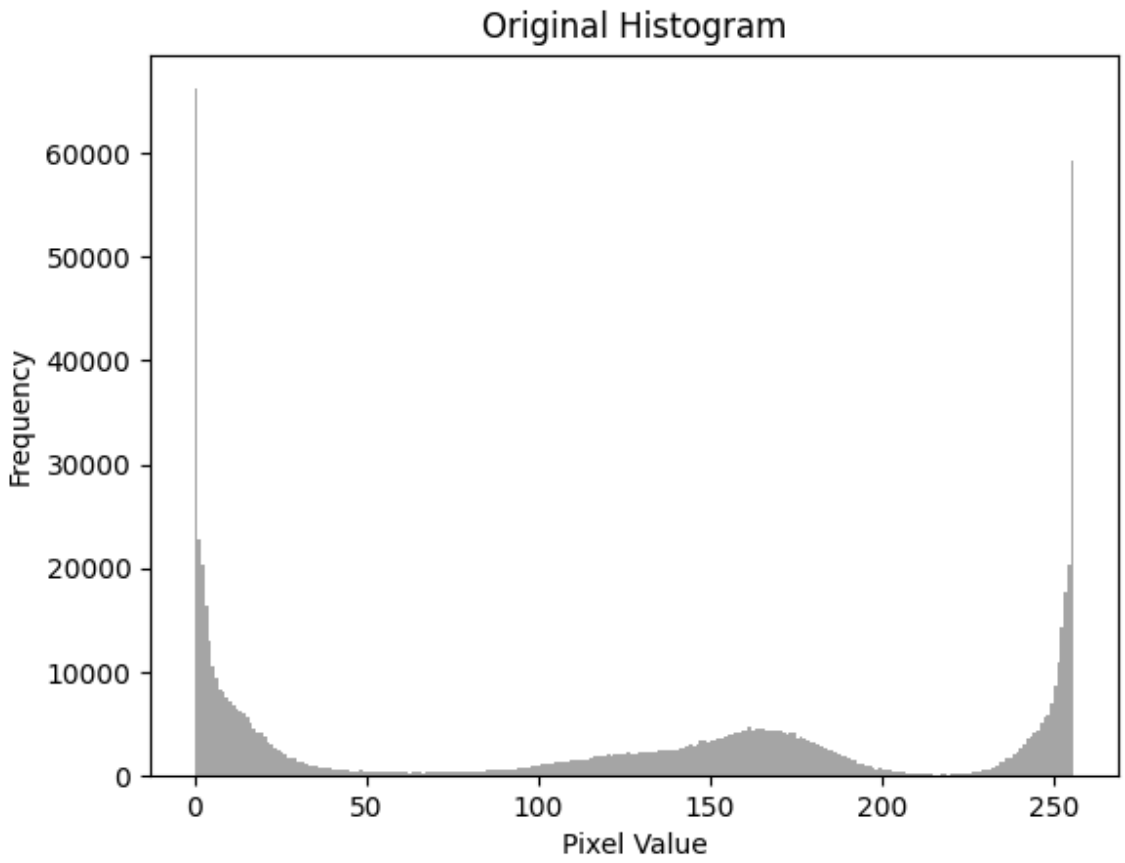
image 90 degrees clockwise. The achieved image is:



- **Histogram equalization:** First we started by loading our original image and plotting its histogram. Our Original image is:

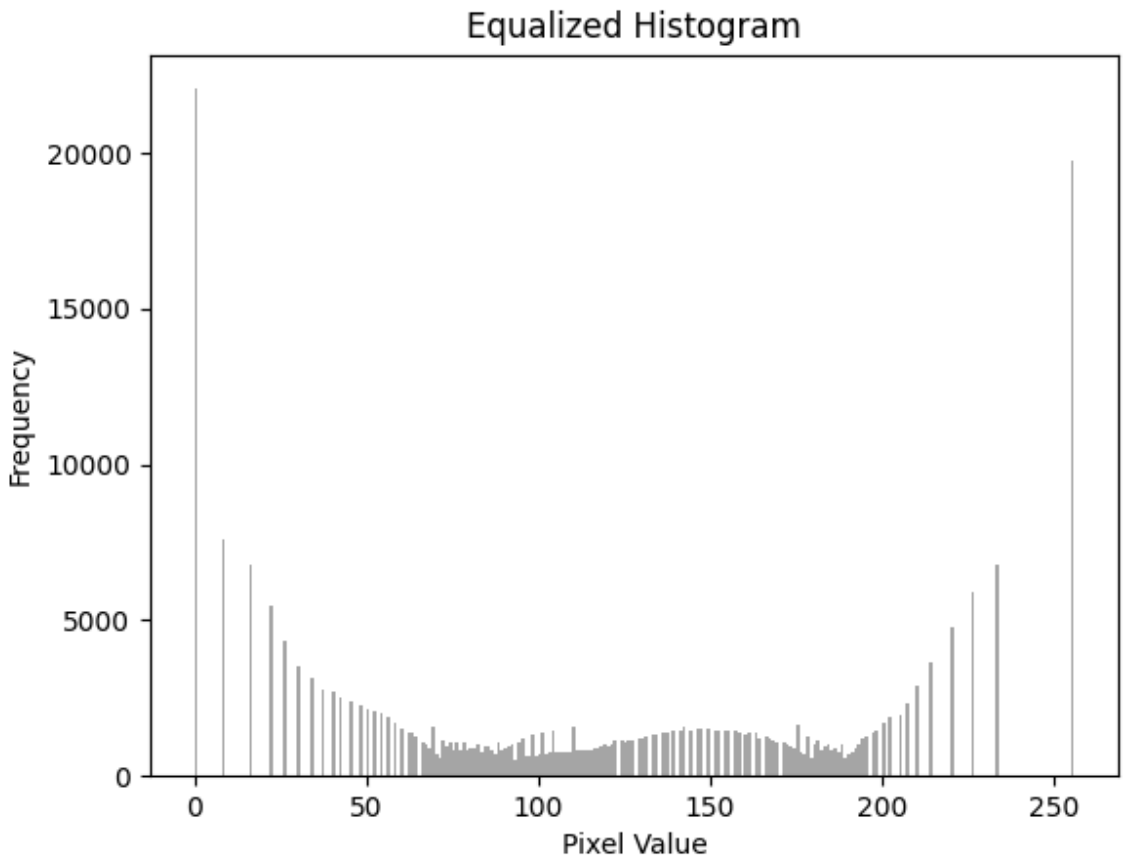


Our original image's histogram is:



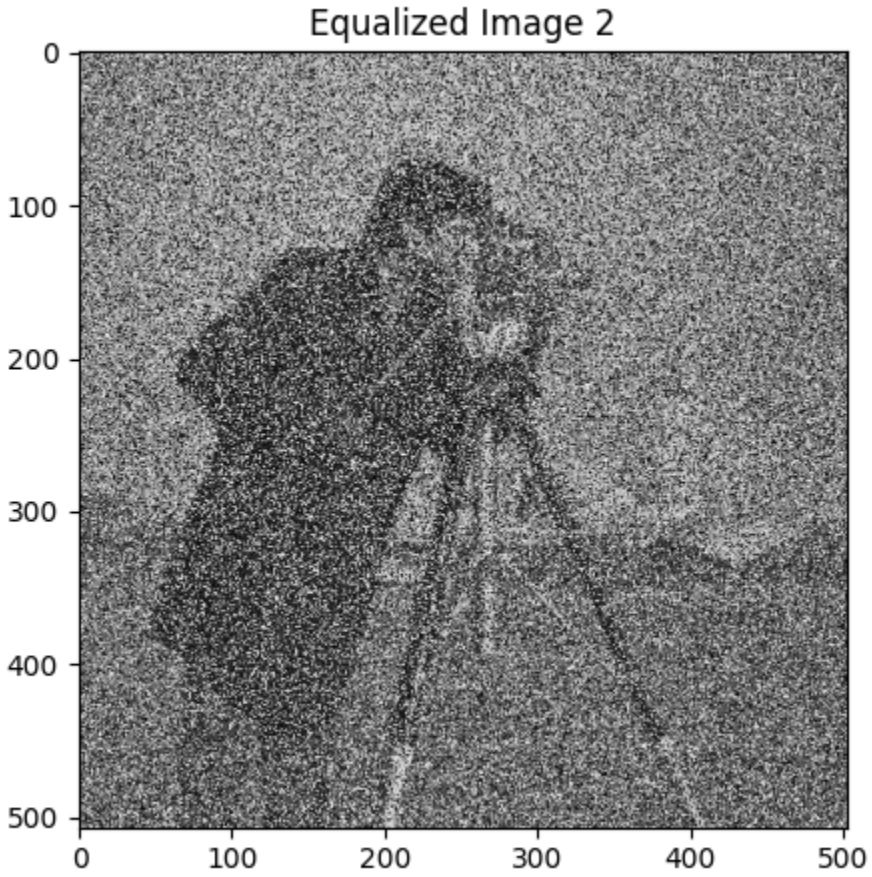
Then we first converted our image to grayscale and performed histogram equalization using equalizeHist of the cv2 library. After histogram

equalization, our new histogram is:



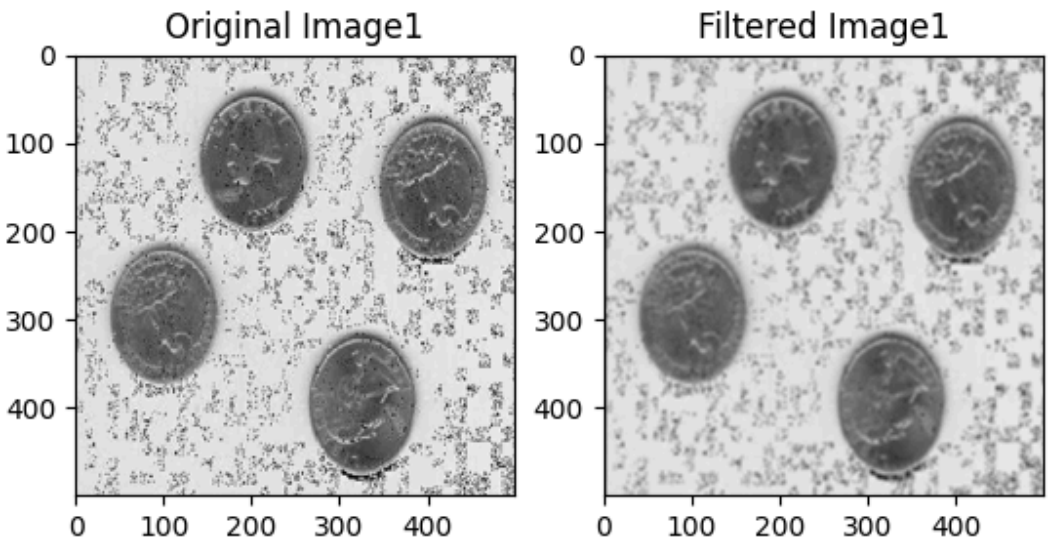
As we can see, because there was a huge difference in very dark and very bright pixels (because of the noises), our new histogram still is not very equal. And again because our noises were misleading our histogram equalization, we can see that our new image has not been able to remove

the noises.



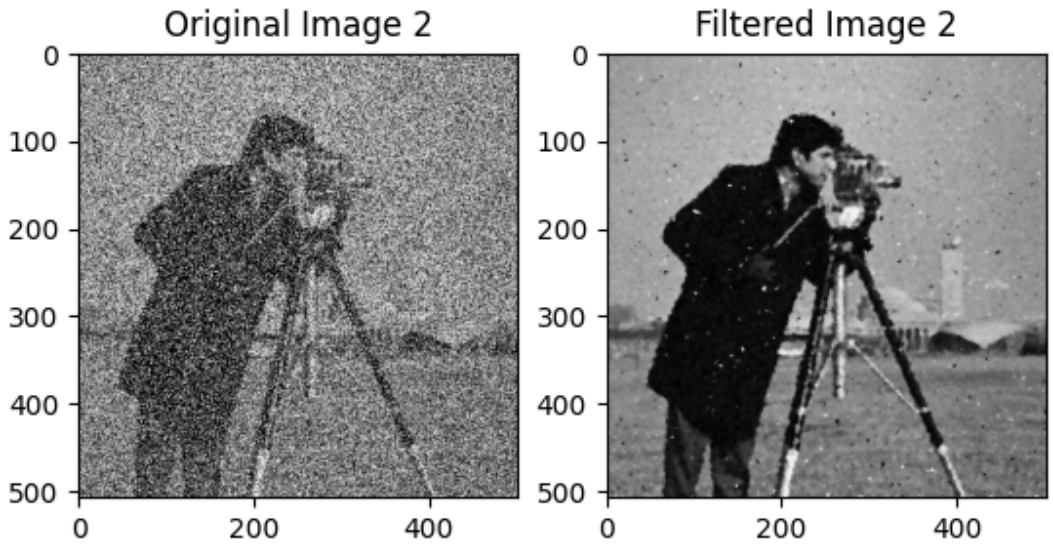
- **Filtering in Spatial Domain:** We started the spatial filterings with the average filter. To do so, we developed a 5 * 5 matrix filled with 0.04, and we used this matrix, as a convolution matrix to apply it on our image. To do so, we used the filter2D function of the cv2 library and the resulting image

beside the original image is:



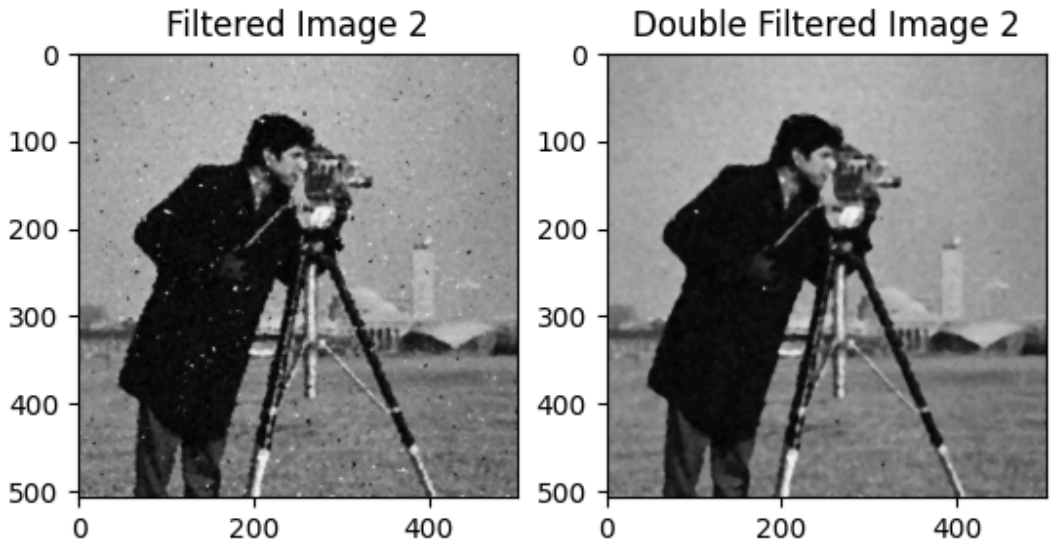
We can see that because of the averaging, the sharpness of the image is reduced and pixels change more smoothly.
After the average filtering, we implemented median filtering, using directly the medianBlur function of the cv2 library with a filtering size of 5*5. The

achieved image beside the original image is:



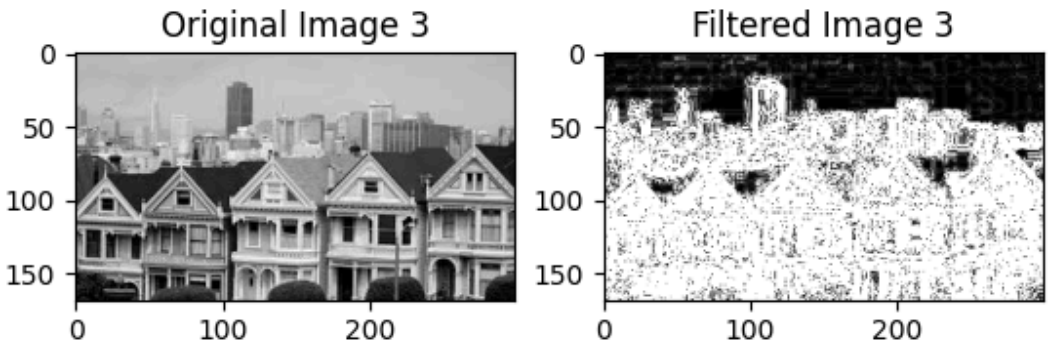
As we can see, instead of histogram equalization, median filtering has been very helpful in clearing the noise out of our original image. The reason is that our noise was a lot of pixels which were very bright or very dark but they were not very close to each other. So in a result, the number of bright and dark pixels is unreasonably high, and histogram equalization will just make them closer to the overall of the image, but in median filtering, the noise pixels are compared to their neighbors, and because the noises are not very close to each other, the neighbors can outstand them, and the noise pixels will be removed. Also, most of the remaining noises can be removed by filtering our image again as we can see in the

double-filtered image beside the filtered image:



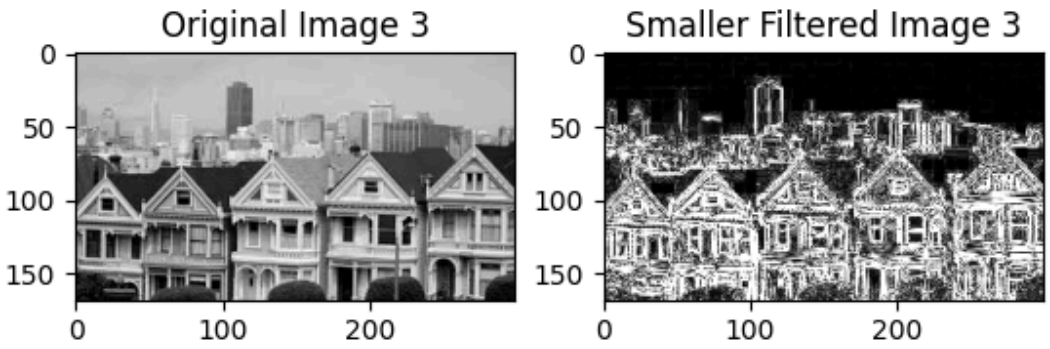
Next, we implemented the Laplacian filtering using a 5*5 filtering size. To do so, we used the Laplacian function of the cv2 library. The resulting

image beside the original image is:



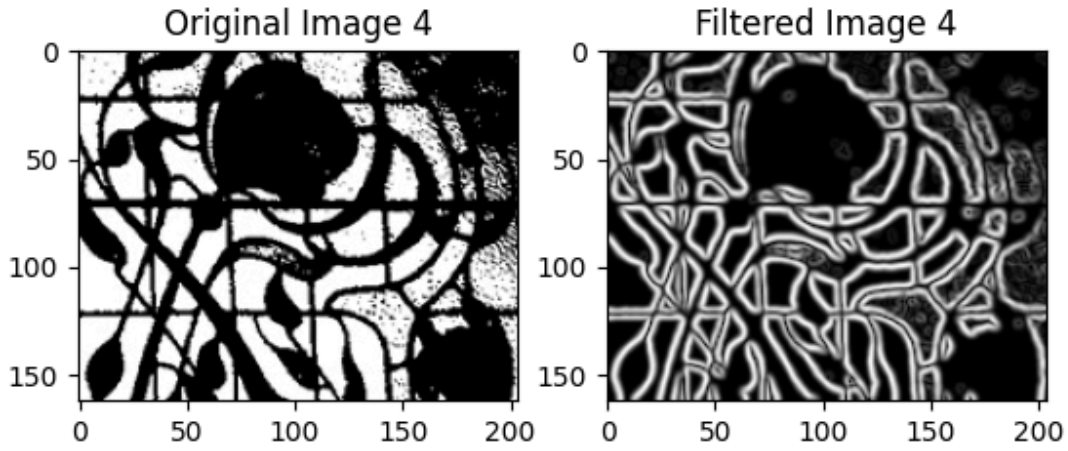
But as we can see, our filtering was not able to detect the edges of the image. The reason is that we have used a large amount of filtering size. To help our filters to find the edges better, we should decrease our filtering

size. For example, if we set our filtering size to be 3, we have:



Now, for the last filtering, we implemented the Sobel filtering with a filtering size of 7*7, both horizontally and vertically, and then we calculated the magnitude of both results as a final result. To do so, we use the Sobel function of the cv2 library to calculate both X and Y Sobel filterings, and to calculate their magnitude, we use the magnitude function of the cv2 library.

The resulting image beside the original image is:



And as we can see, it has been able to find the edges very well.

Key findings:

- **With image translation, there is a risk of losing data**
- **Histogram equalization is not able to clear random black-and-white noises**
- **Average filtering smoothes the image**
- **Mean filtering can remove random noises**
- **Performing filtering multiple times has different outputs**
- **Laplacian filtering needs small filtering sizes**

References:

- **OpenCV (cv2) library (for most of the calculations)**
- **Matplotlib library (for visualizing images and histograms)**
- **Numpy library (for image translation matrix)**