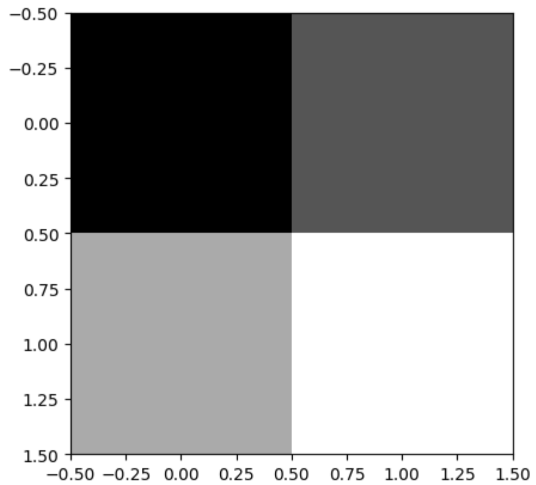
Report- Assignment 2

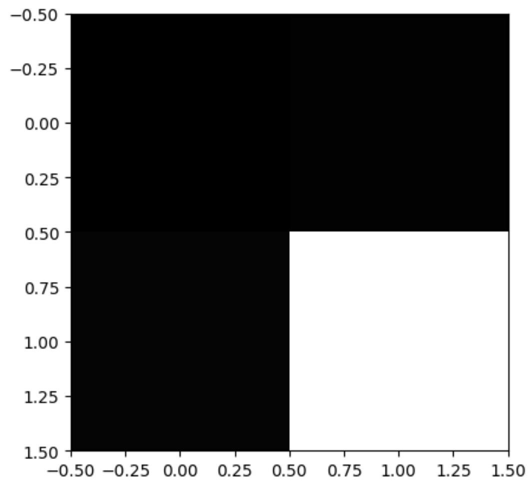
1. **Image Display: When the image 'ECE.png' (Figure 1a) is displayed with the following code:**
   import matplotlib.pyplot as plt
   from scipy.io
   import imread im = imread('ECE.png', cmap='gray') plt.imshow(im)
   plt.show(),
   we get a result shown in Figure 1b. Explain why there is a difference between the actual image and the displayed image. Find out a way to prevent this from happening. Note: This question has to be compulsorily solved in Python.

   Inference:

```
plt.imshow(np.array([[1,2],[3,4]]), cmap="gray")
plt.show()
```




```
plt.imshow(np.array([[1,2],[3,99]]), cmap="gray")
plt.show()
```



- We can see in the images that a 2x2 array is printed the image on left has highest number 4 and the one on the right has 99 as the highest.
- If we change the highest number in array it makes the 1, 2 and 3 pixel intensity become darker. This means that the image is being full scale contrast stretched by the plot function by default.
- We can search for the function parameters of imshow of matplotlib or any other function that is being used to stop this from happening.

matplotlib.pyplot.imshow

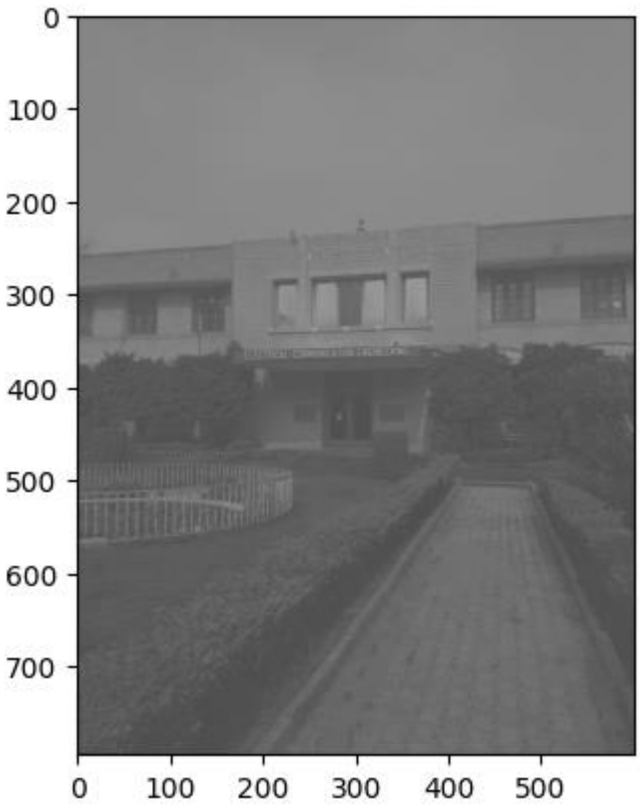
```
matplotlib.pyplot.imshow(X, cmap=None, norm=None, *, aspect=None,
interpolation=None, alpha=None, vmin=None, vmax=None, origin=None,
extent=None, interpolation_stage=None, filternorm=True,
filterrad=4.0, resample=None, url=None, data=None, **kwargs) [source]
```

vmin, vmax : ***float, optional***

When using scalar data and no explicit *norm*, *vmin* and *vmax* define the data range that the colormap covers. By default, the colormap covers the complete value range of the supplied data. It is an error to use *vmin*/*vmax* when a *norm* instance is given (but using a `str` *norm* name together with *vmin*/*vmax* is acceptable).

This parameter is ignored if *X* is RGB(A).

```
im = io.imread("/content/ECE.png")
# show_histo(np.round(im))
plt.imshow(im, vmin=0, vmax=255, cmap="gray")
plt.show()
```



2. Contrast Stretching:

(a) Write functions to apply histogram equalization and power-law (gamma) transforms on a greyscale image.

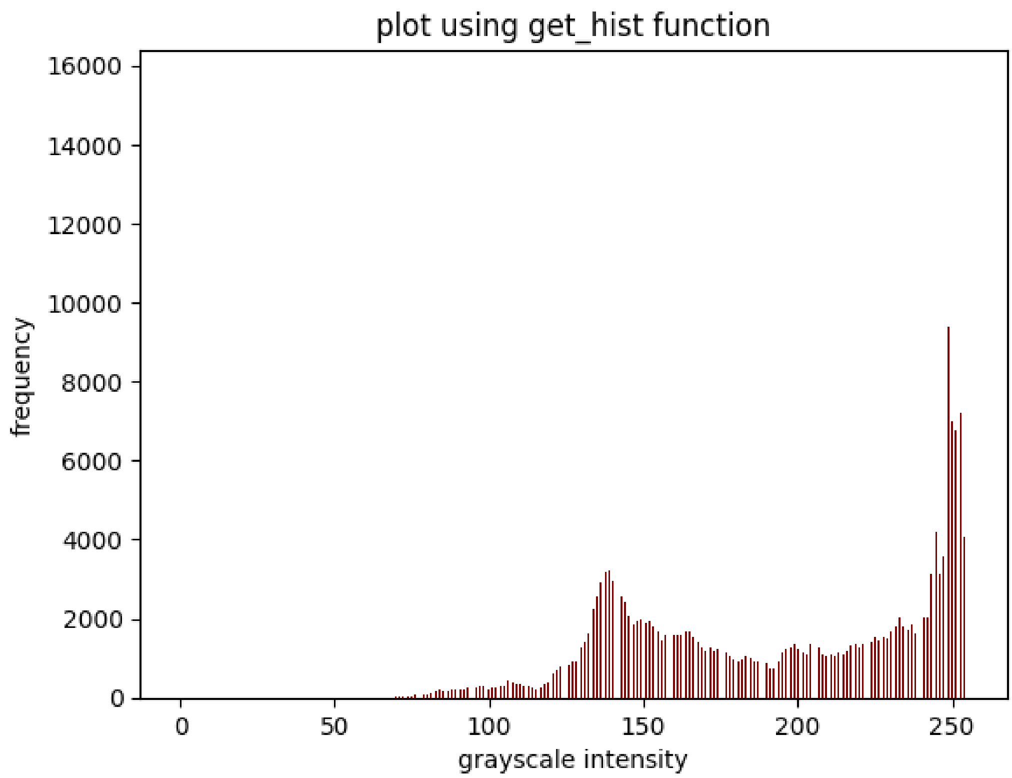
(b) Apply histogram equalization on 'haze.png'.

(c) Power-law (gamma) transformation: J(i, j) = I(i, j) γ . Find the optimal parameter γ ∈ (0, 5] that minimizes the mean squared error between the histogram equalized image from (b) and the gamma transformed image. Use this γ to transform and display the input image.

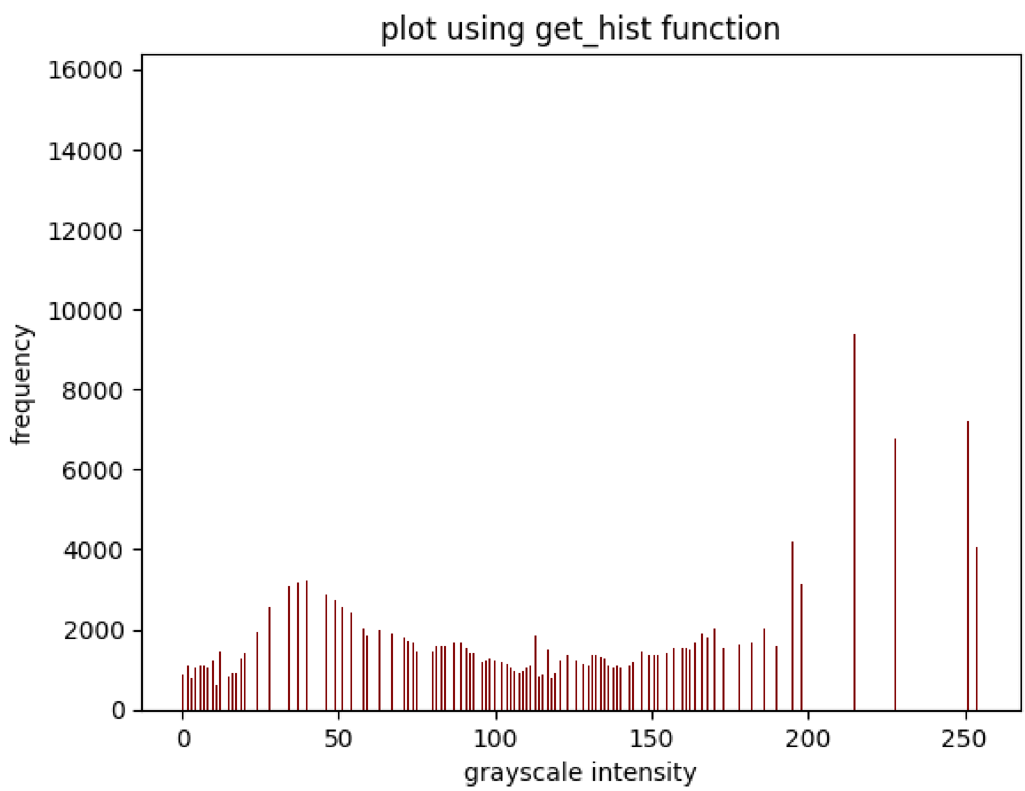
Visualize the histogram before and after each operation and comment on how they relate to the resulting images.

Report:

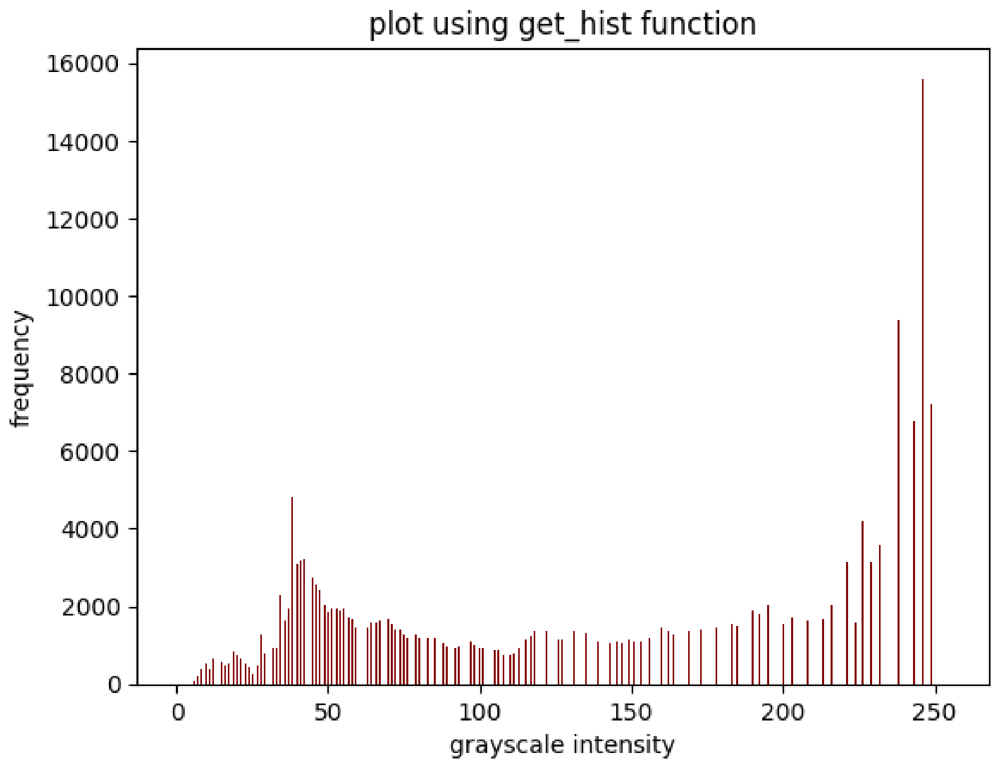
The histogram plot before image processing,



The histogram plot after Histogram equalization,



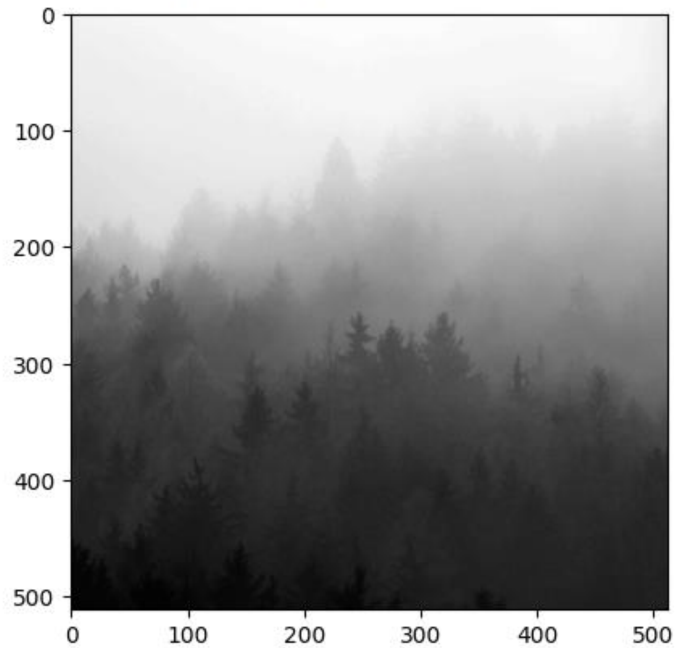
The histogram plot after NLCS,



The images of Histogram equalization(former) and NLCS (later),

The minimum mean square error of the two functions and gamma value = 3.01

```
76176671 3.01
<matplotlib.image.AxesImage at 0x7cd41c2476d0>
```



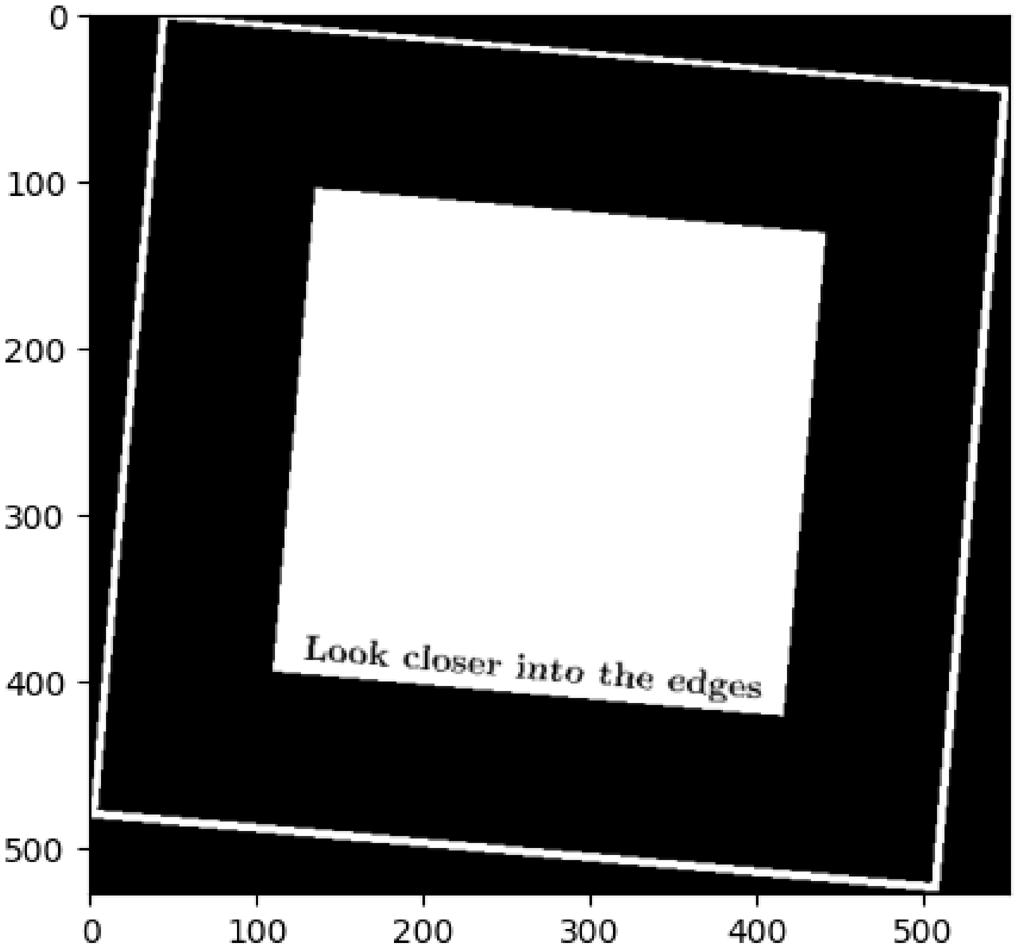
Inferences:

- In the histogram of NLCS we can observe the number of higher intensity pixels have increased.
- The histogram of the Histogram equalised image is more evenly spread out and the peak of the histogram remains in the same value. Which is not the case in the NLCS image.
- The Histogram equalised image brings out more details of the image like the hills in the background because of the more well-spread histogram
- But in NLCS that is not the case because the higher pixel value increases and the lower values stay lower due to this some of the details are lost.

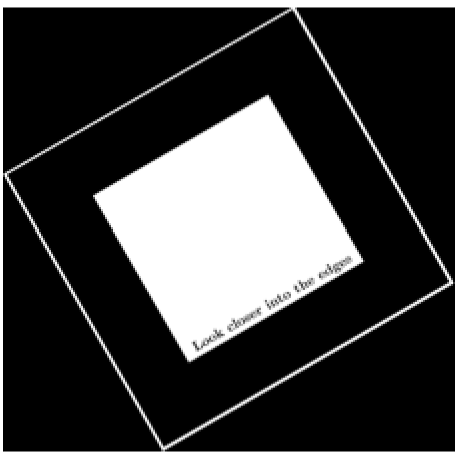
3. Image rotation: Write a function to rotate the image 'box.png' in the clockwise direction by 5 ◦ and counter-clockwise by 30◦ using both nearest neighbor or bilinear interpolation. Make sure the rotated image is completely visible and not cropped. Observe the edges of the first image after rotation and comment on the difference between each interpolation method.

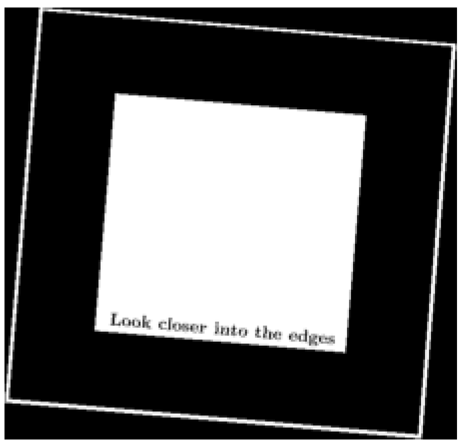
Report:

Rotated Image using interpolation by nearest neighbour method.

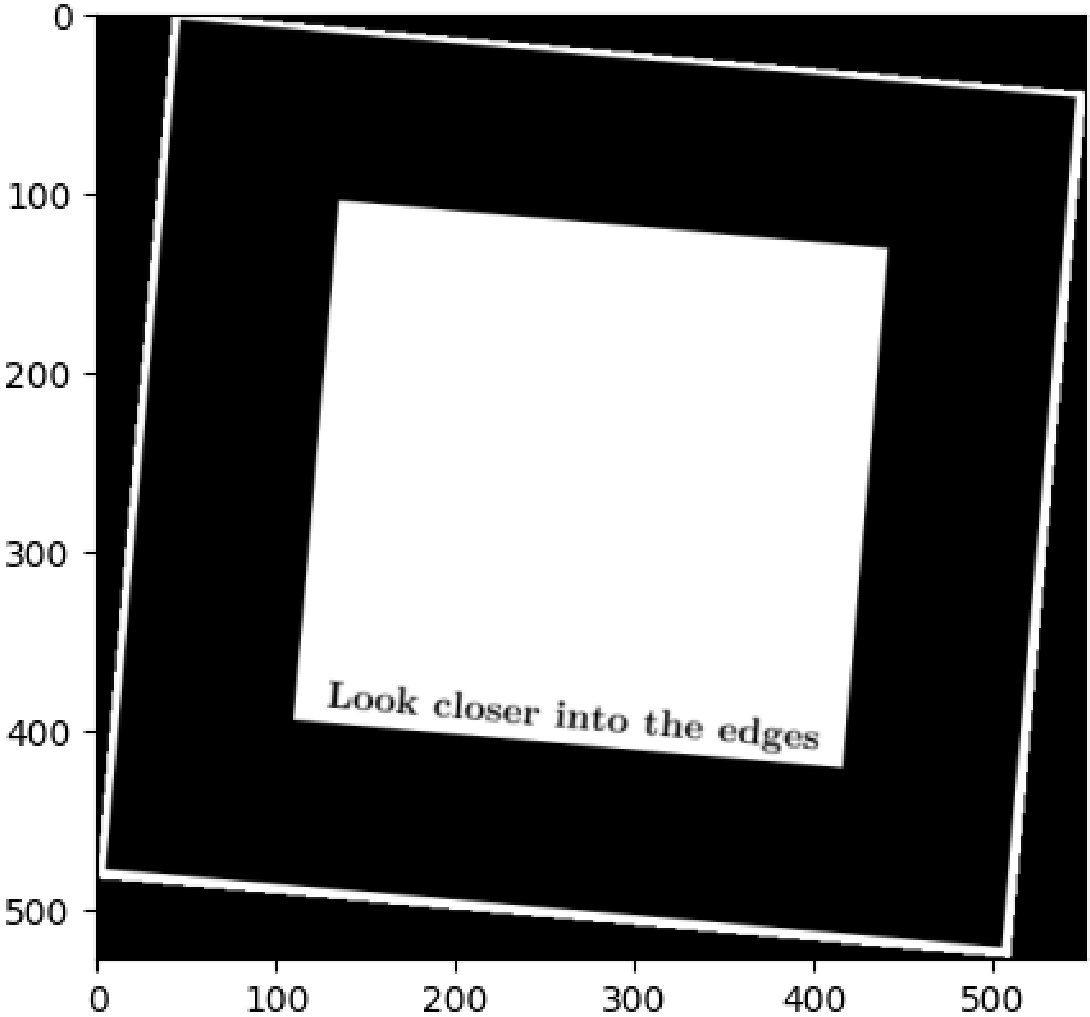


Results (nearest neighbour method)

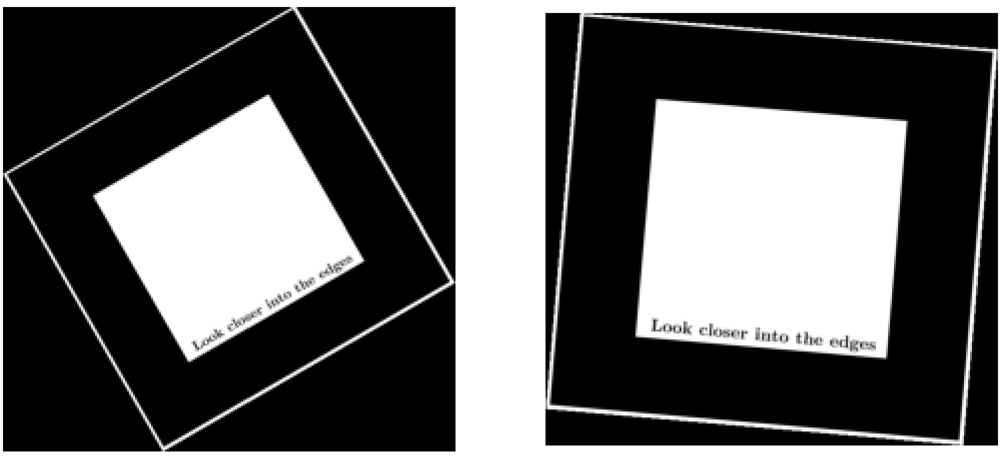




Rotated Images with interpolation using (close view) bilinear interpolation.



Result (bilinear interpolation)



Inferences:

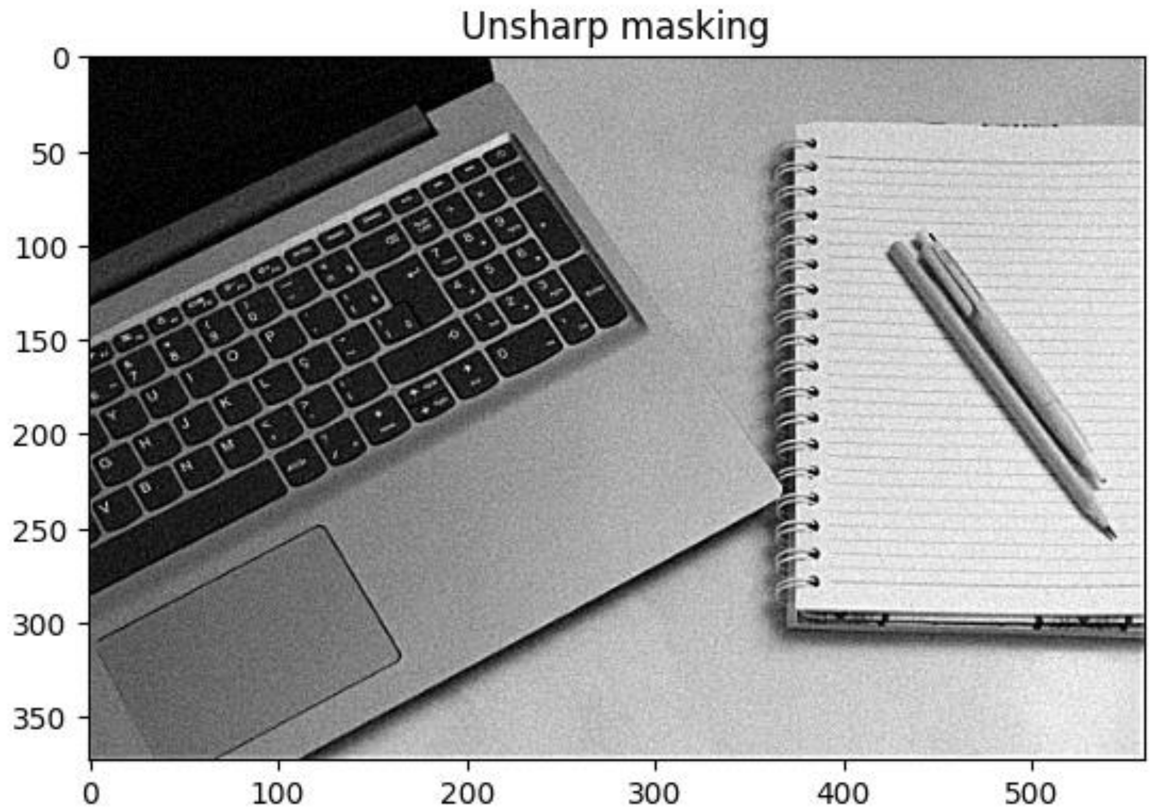
- We can observe from both the images the bilinear interpolation is obviously a better choice.
- In the nearest neighbour implementation the image is more distorted, it can be seen clearly from looking at the edges of the image, the letters written in the centre also seem more distorted in the case of nearest neighbour.
- The reason for this can be understood from the underlying logic of the operations, where nearest neighbour is just taking the rough estimate of what the value each pixel might take the bilinear interpolation takes account of what all the nearest pixel around a given point are.
- This implementation helps to look deep and close into the working of interpolation using a simple rotation mapping. We were supposed to rotate the image coordinate ground and map the original image pixel values to the rotated coordinates using some interpolation technique.

4. Spatial Filtering: Apply high-boost filtering on the image 'study.png' to sharpen it. Use square averaging of size 5 to perform unsharp masking. Choose the scaling constant for the high pass component as k = 2.5. Comment on what you observe. Now, blur the input image with a square average filter of size 3 and perform high boost filtering again. Comment on the difference in output compared to the previous case.

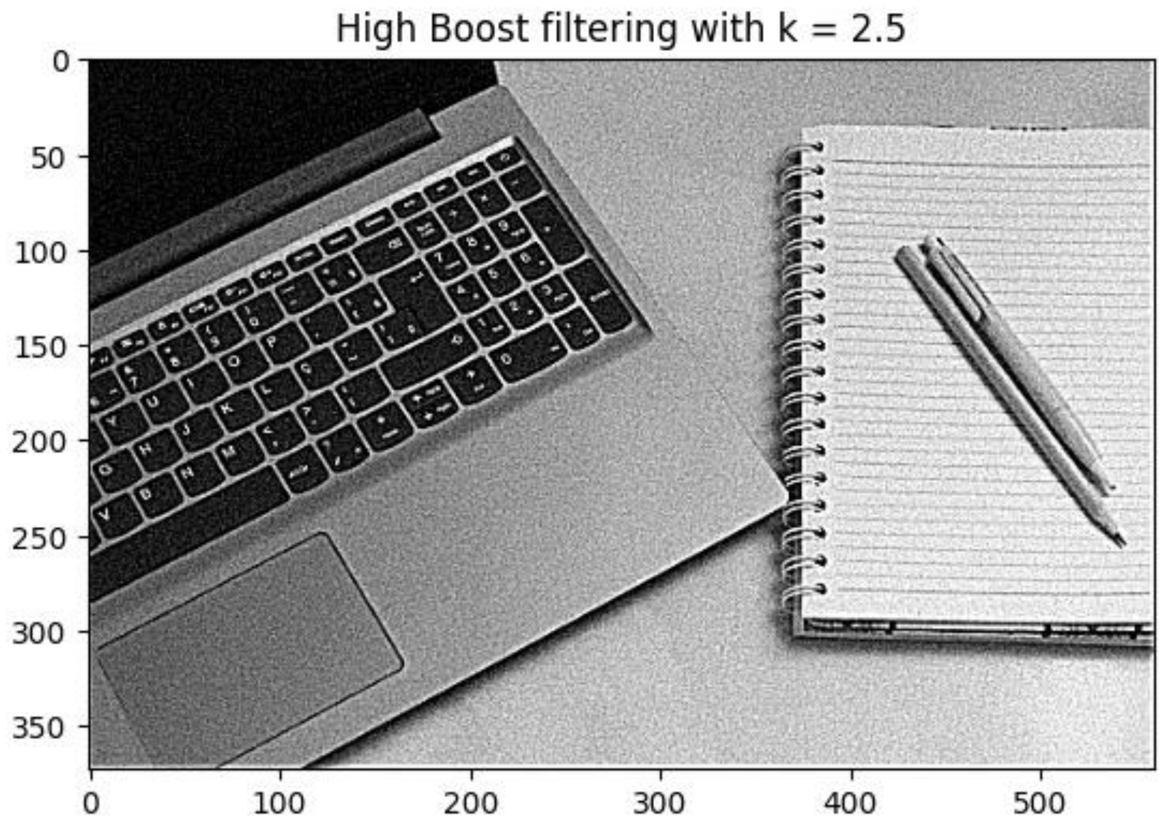
Report:

Original image,

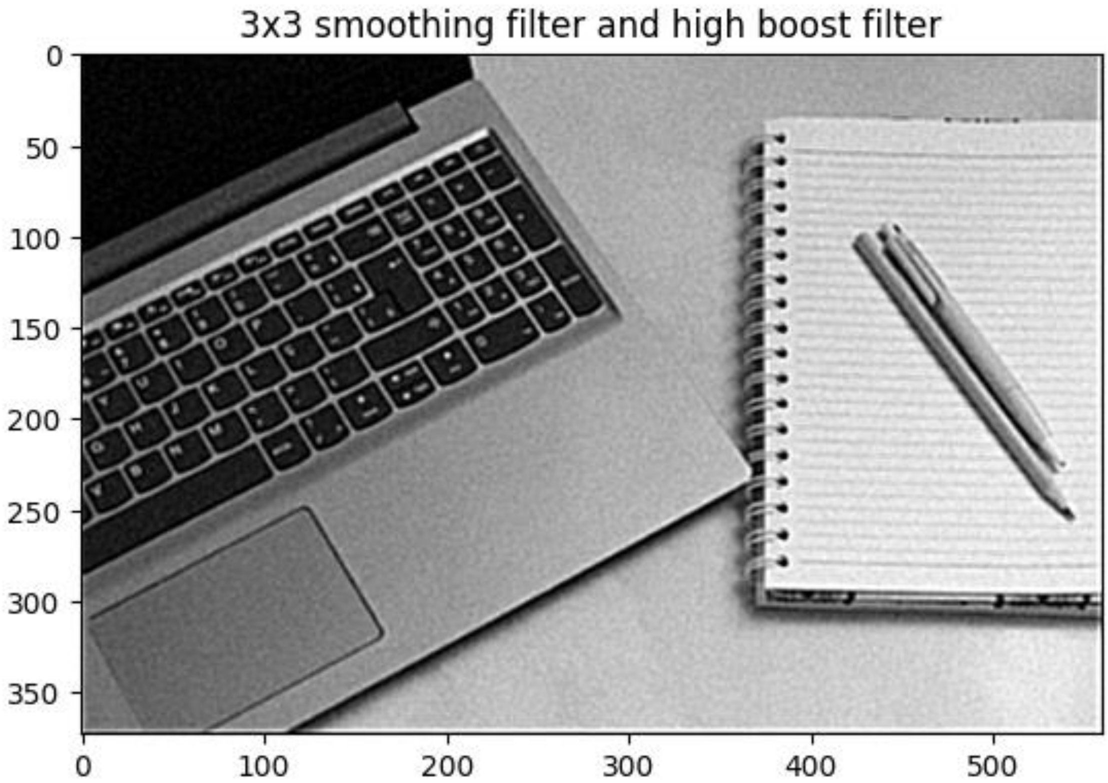
Unsharp masked image using 5x5 square averaging filter,



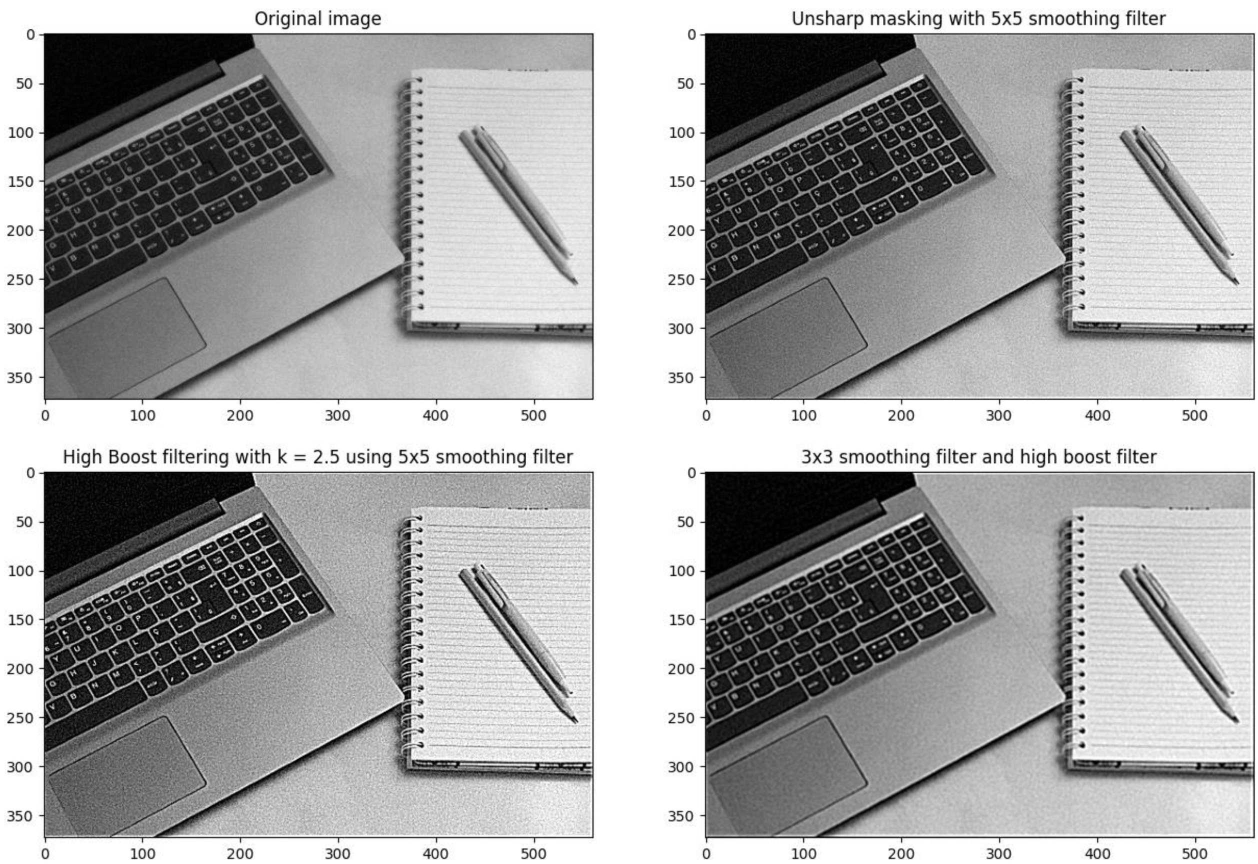
High boost filtering with 5x5 smoothing filter,



Using 3x3 LPF before high boost filtering with high pass filter component k = 2.5.



Results



Inferences:

- First thing that can easily be observed is that the high boost filtering with 5x5 smoothing filter is the sharpest.
- But the image is more noisy, that is the boost filter has also caught some noise in the process.
- There is some degree of smoothness to the image which is subjected to LPF(3x3) before performing high boost. It also has less noise compared to the unsharp masked image.
- The reason for this difference can be explained by intuition, that is the 3x3 LPF filtered image has less of the high frequency components than the normal image this reduces the noise in the input of the sharpening filter thus reduces noise compare to the others.
- This also results in the image being more blurred and the details of the keyboard are less visible.