Question1:

Image1: Average Filter of the values of the line

Since we can see that the image has been transformed to be completely blurred but noticing that the color definition of each row is kinda the same as the original image color we can deduce that this is row based filtering where each row maintained the colors but the shape of the objects has dissipated. And I have written the code for it and here is a comparison between image1(left) and my reconstructed image(right) based on average. MSE=0.5

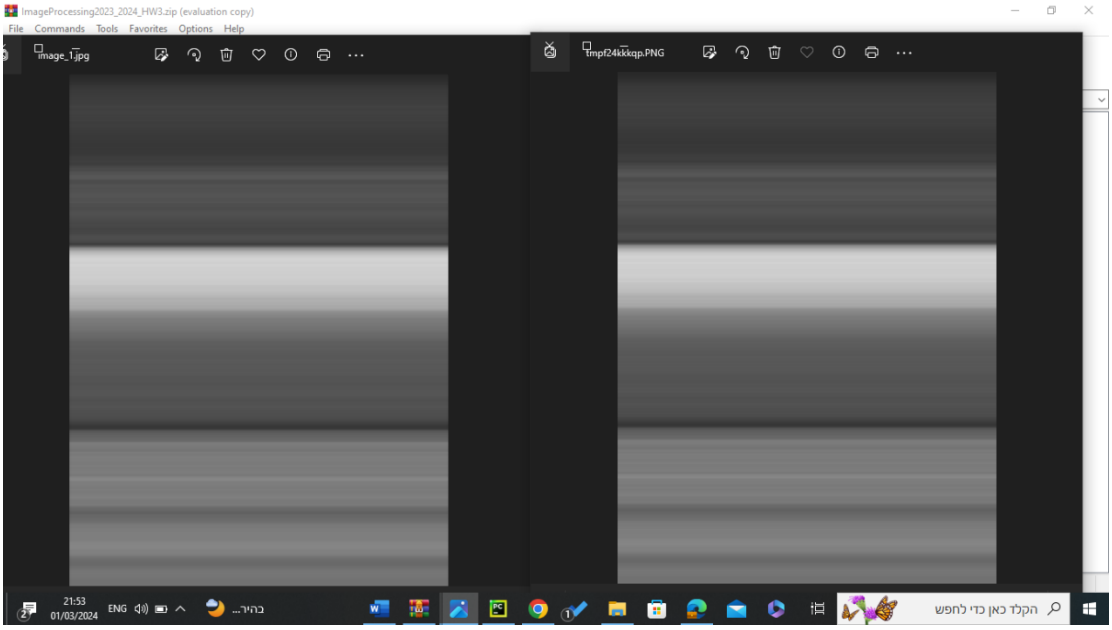


Image2: Average/Mean/Smoothing Filter

I chose this because it is blurry and as we have seen in tutorial 5 this is caused by the Average filter. Also there is a little white lines on the edges of the picture.

I have written the code for it and we can see comparison between image2(left) and my reconstructed image(right). MSE=2.7

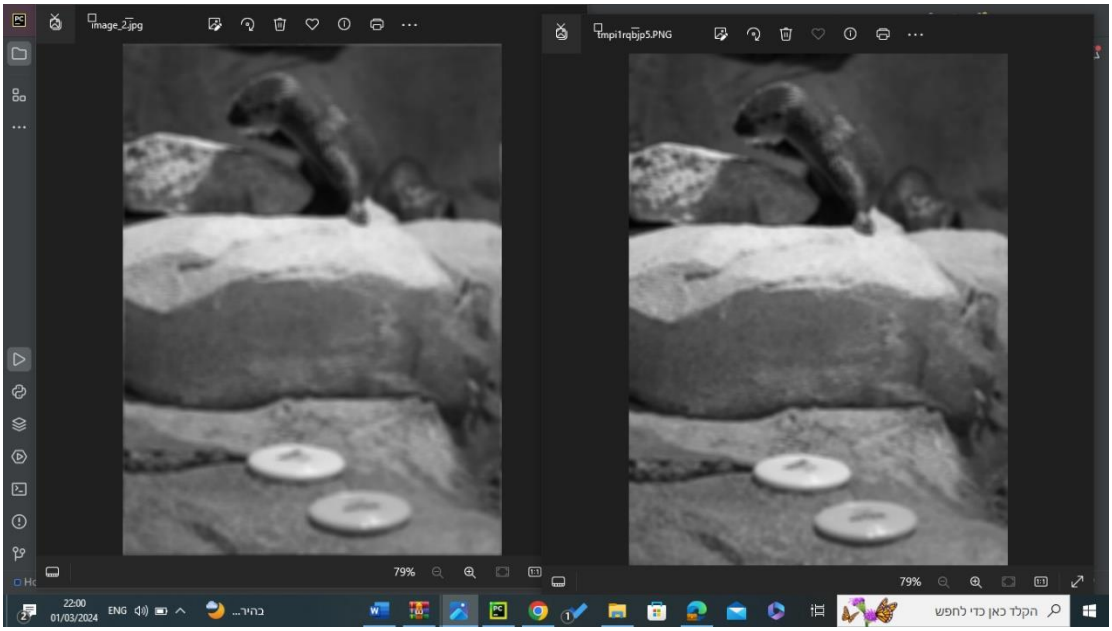


Image3: Median filter with radius=9

I have deduced that using a median filter with radius=9 will get me to the correct filtering for this image. I chose median filter because the image seemed smooth and nice and that's kind of what happens when we use median filter.

Image 3 is on the left and my constructed image is on the right. MSE=4.2

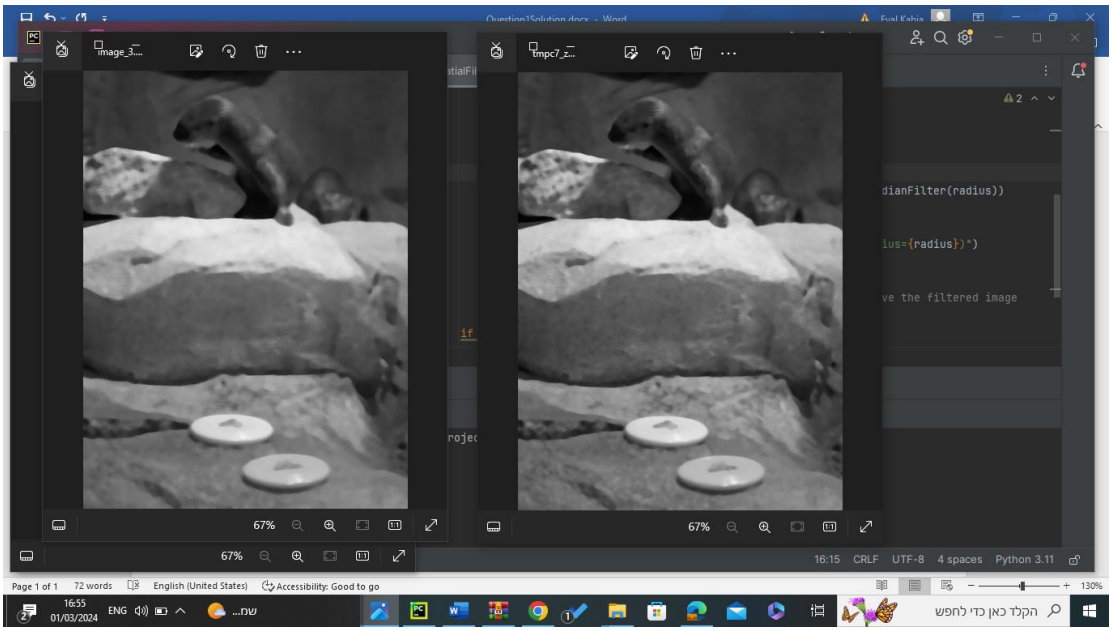


Image4: Average Filtering- shaky vertical blurring

I choose average filtering because it seems like the image is shaky which means that the image has been altered in such a way that each pixel takes from its surrounding pixels the average value and takes it , but we have to pay attention that The size of the kernel is specified as (1, kernel_size), indicating that the filter operates only vertically with a kernel size of kernel_size pixels in height, that is what gives us the shaky effect also here is a comparison between image4(left) and my reconstructed image(right).MSE=4.7

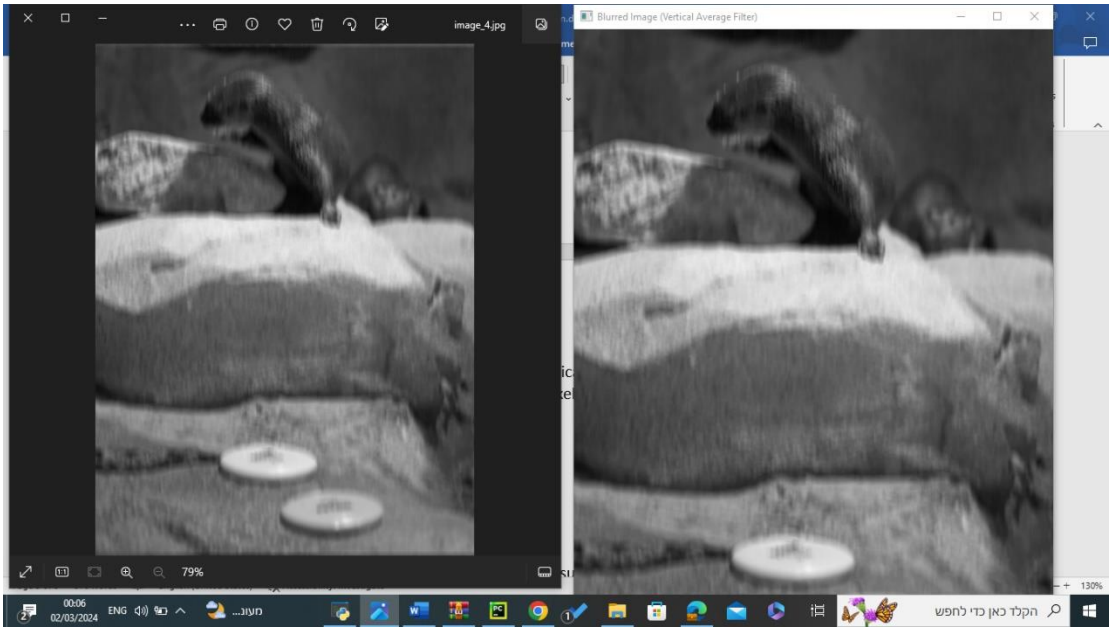


Image5: Sharpening-gausian

We chose gausianl filter because it gives us the best Sharpening effect , which is basically what we are seeing in image 5, because the edges around the objects are more sharp despite the blurriness in the image that is shown on the otter and the rocks.(their edges are sharp nonetheless)

This is visual representation of difference between image5 (left)and my reconstructed image(right).MSE=4.8

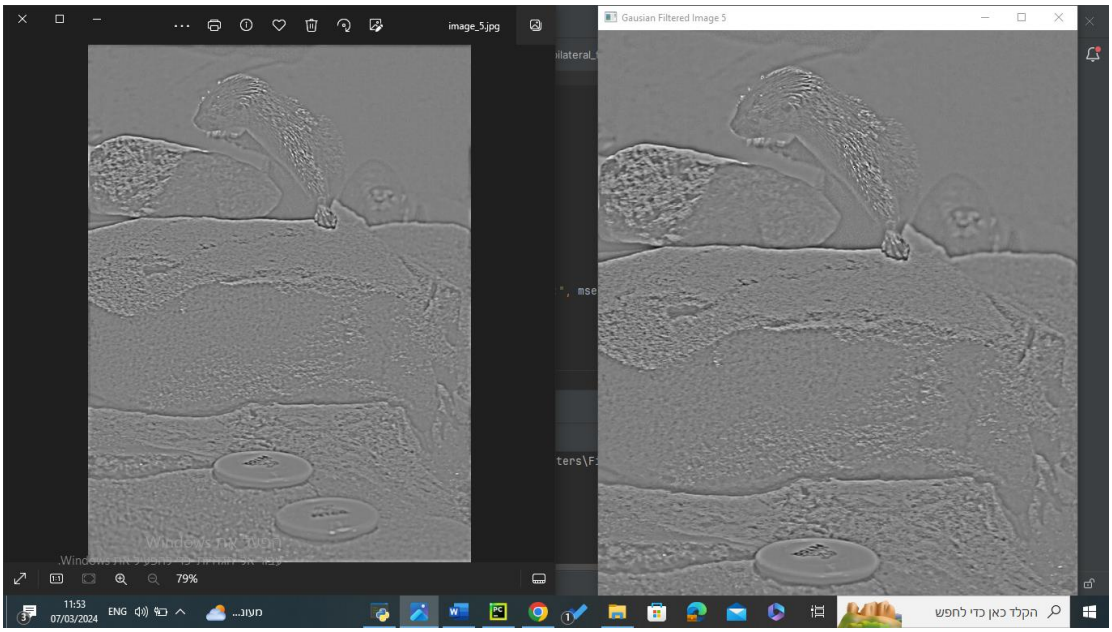


Image6: Laplacian Filter

In here I decided we need to use the Laplacian filter because as we have seen in tutorial 5 we use Laplacian filter to show the edges better so we can define the shapes of the objects in the image. I have done the code to show the reconstructed image after doing Laplacian filter and this is the result: image6 is on the left and my reconstructed image is on right and the MSE=5.2

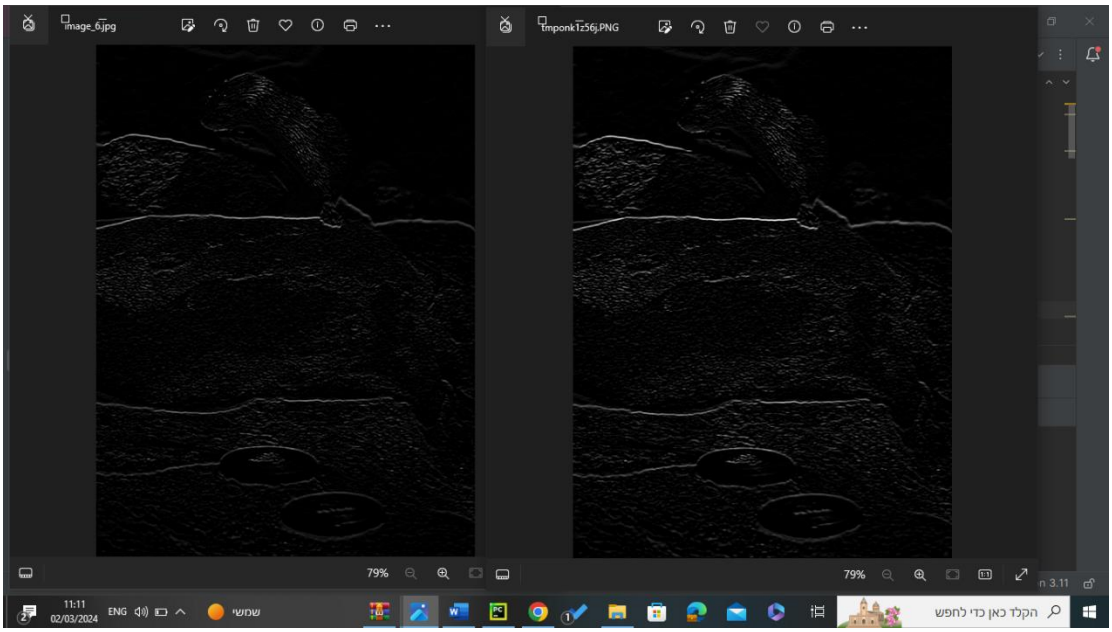


Image7: Vertical Flipping-Wraparound

As we can see the image is flipped vertically such that the bottom half is now up and the upper half is now on the bottom. This can happen as a result of wraparound where u make copies of the original image and then shift the values up or down(depending if the copy is above or below the image) After reconstructing I have come to this image.

Image7 is on left and my image is on right. MSE=0.003

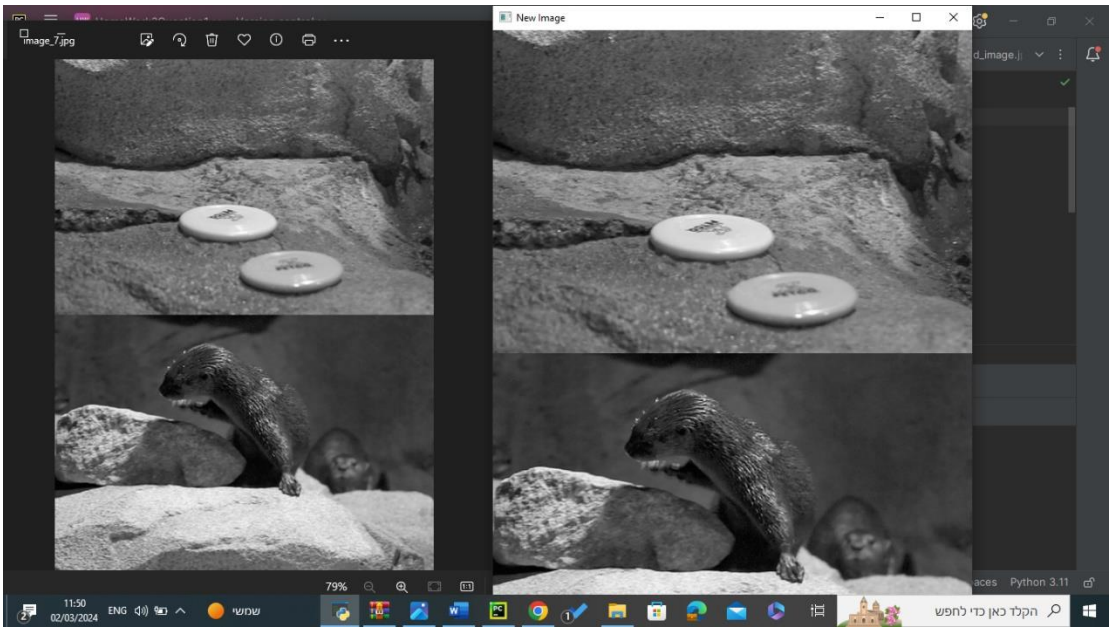


Image8: No filtering was done

As we can see it is basically the same image just after it was gray scaled. On the left we see image 8 and on the right we see the gray scale image

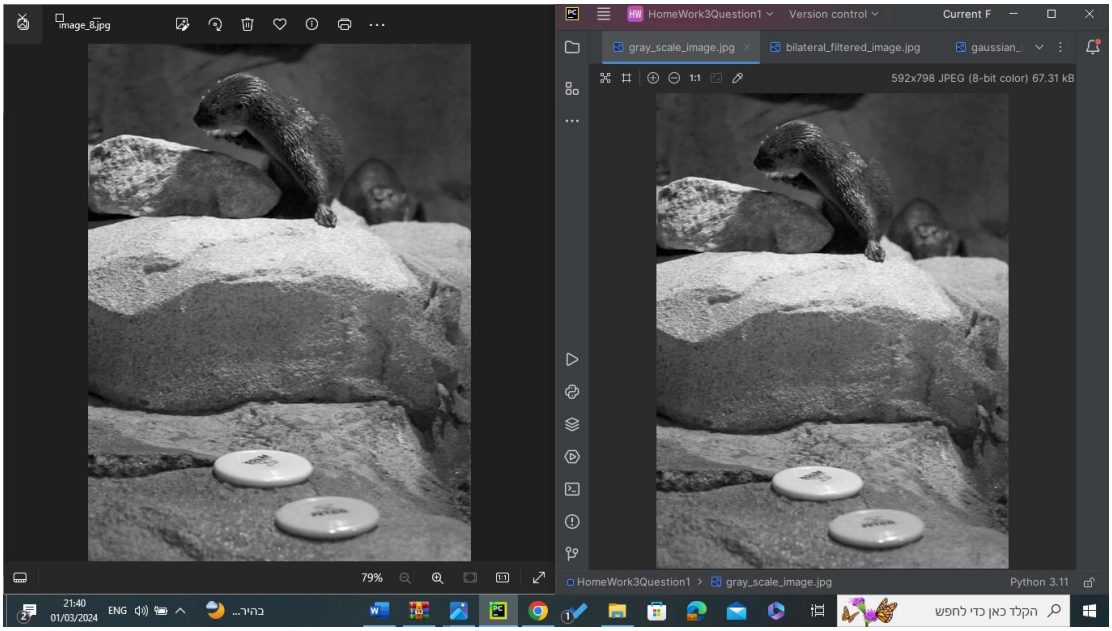


Image9: Sharpening-Laplacian Sharpening

As we can see the image seems enhanced and sharpened so that the features are more visible and the contrast between the colors seems more defined. In the left we have image 9 and on the right we have our reconstructed sharpened image. As u can see they are almost identical. MSE=10.9

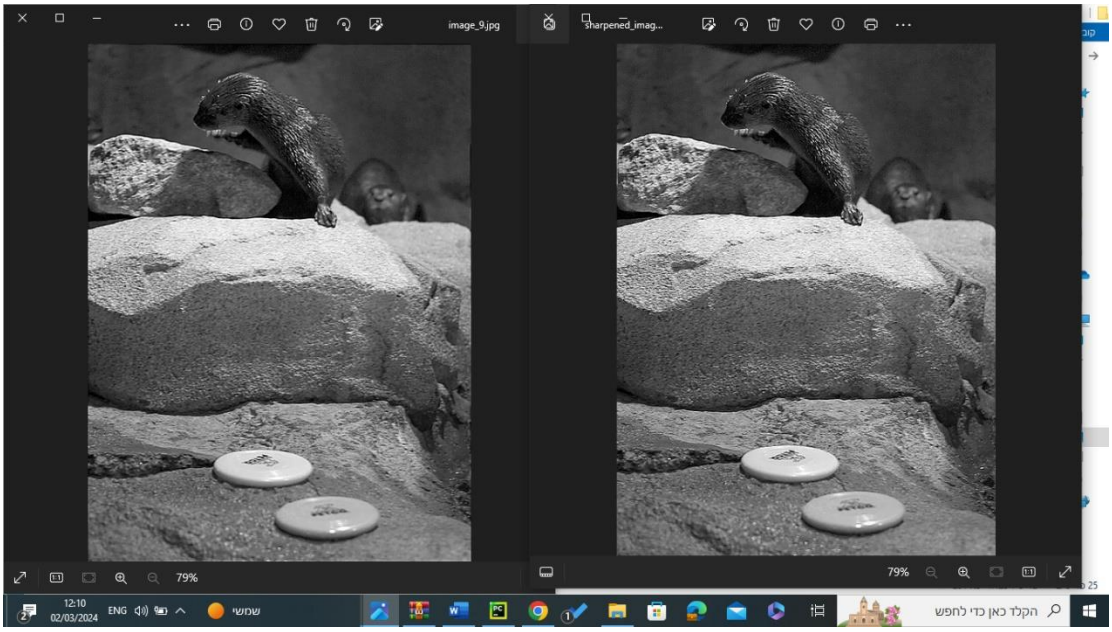


image_9.jpg
sharpened_imag...
79%
79%
12:10
02/03/2024
ENG
שמשי
הקלד כאן כדי לחפש

Question 2:

Code explanation:

We will 2D array of the X,Y(x represent the x direction the same for y) coordinates of a grid from the range of (-radius – radius) and it will be used for applying spatial calculations over the window that

We will center around each pixel.

After that we will define gs which is matrix of gaussian spatial weights

And this weight is based on the distance of the center of the window

gs = np.exp(-(x2 + y**2) / (2 * stdSpatial**2))**

so, each we need to compute it we will just give it the values of x and y

define our window as given:

$$window(i,j) = im[i - radius: i + radius + 1, j - radius: j + radius + 1]$$

Taking in consideration to not get out of the boundaries.

Calculating gi gs for each I,j in the window as asked using

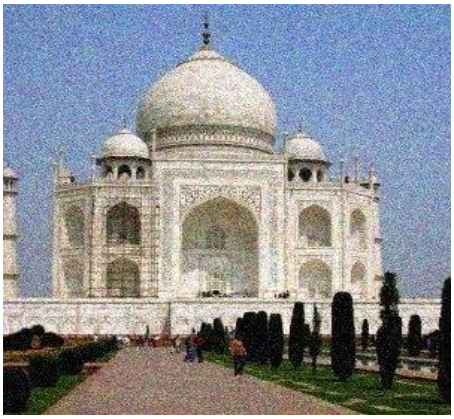
Np.exp

At the end we will put the result of the pixel[I,j] in the res_img

b)

images:

taj image:



as we see in the image tag there is a lot of spaces that has the same thing like the sky…

so knowing that we want a larger range of colors to influence the outcome will pick a big intensity parameter

also I choose a big window [9*9] to take a look at big amount of colors around so get the best cleaned picture and also I take the intensity using the function np.std(image) which calculate the amount of noise in the picture I choose at the end 30, getting the spatial parameter by doing 2% of the image diagonal.

In my opinion this image can be properly cleaned using the bilateral filter because it will preserve the edges such for this image there is a lot of edges.

The final result :

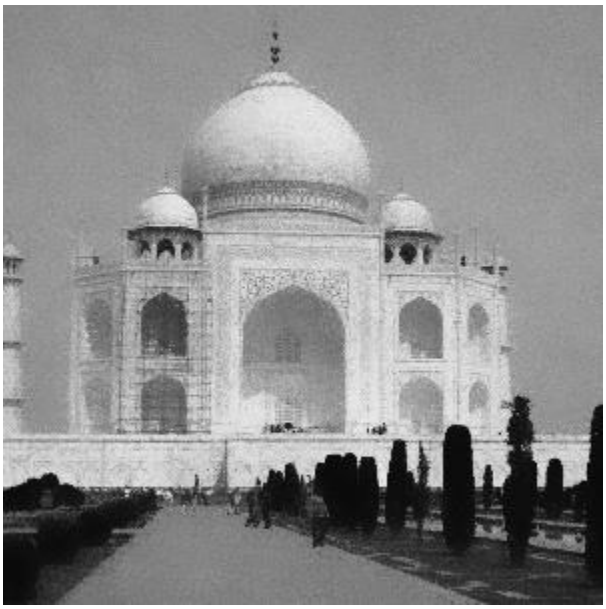
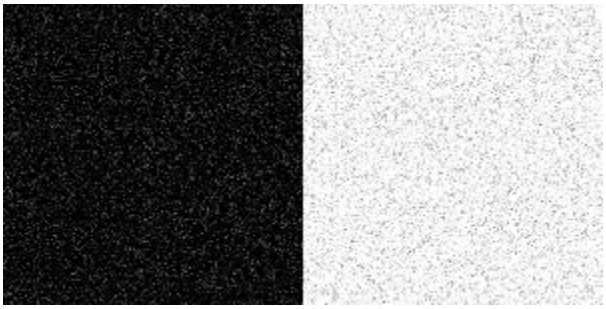


Image NoisyGrayImage:



In this picture as we can see that there is some noise like salt and Baber noise

In this picture I want to get back to the original picture which is half black half white

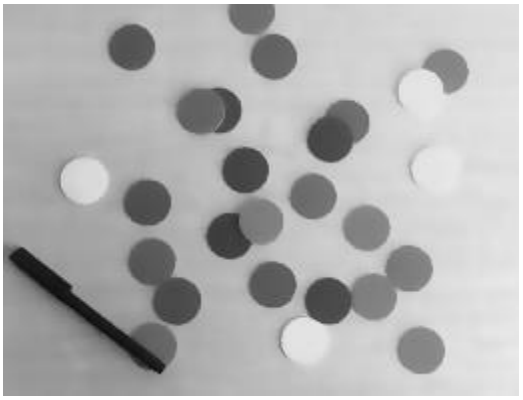
And as we can see there is a loot of noise from that fact, we will need a large window [9*9]

Also a large intensity parameter (so if we where in the white side we will try to influence the pixel

With a lot of white pixels and the same for black) value of 100, getting the spatial parameter by doing 2% of the image diagonal.

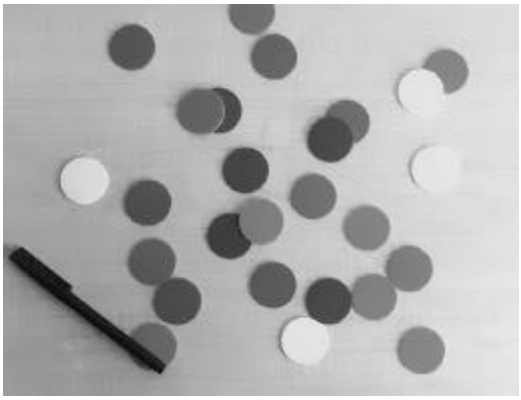
In my opinios this picture in not the best for bilateral filter because we what to consider the Colors that going to melt together due to the split line and the noise.

Balls image:



Here there is a small loss of information by we managed to smoothen the picture in the other hand

There is this picture with less information loss but we can still see the noise



The filter wase useful because it enhanced the edges and cleared the small noise without losing a lot of details

The parameters used for the three images :

img_path_list = ['balls.jpg', 'taj.jpg', 'NoisyGrayImage.png']

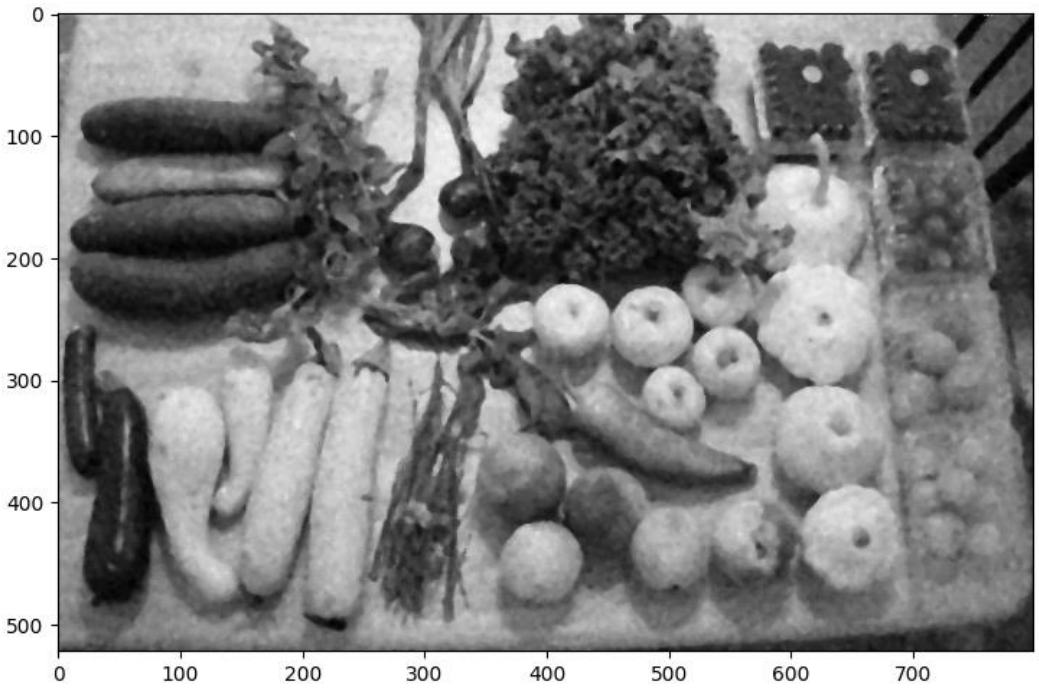
list_spatial = [6.47, 8.49, 6.87]

list_intensity = [20, 30, 100]

list_radius = [3, 9, 9]

Question 3:

a) To clean the broken image we will use bilateral Filter to smooth the edges and median filter to
bilateral_img_0 = cv2.bilateralFilter(broken_img, d=3, sigmaColor=50, sigmaSpace=19)
sigmaColor = 50 as we estimate them before np.std(median_img1) and it give a good result
sigmaSpace = np.sqrt(broken_img.shape[0]**2 + broken_img.shape[1]**2)*0.02 also I used this from the previous question
and we did a median filter in size 5 because it clean the image much better that 3
clean the rest of the noise this what we will get:



Now let's try to sharp the image more to get more defined edges:
Now to do the sharpening doing it as we do in the lecture :
$edge - enhancement - img = B + \lambda B_{sharp}$
In choose $\lambda = 0.5$,G=$\begin{bmatrix} 0 & \frac{1}{6} & 0 \\ \frac{1}{6} & \frac{2}{6} & \frac{1}{6} \\ 0 & \frac{1}{6} & 0 \end{bmatrix}$ as in the lecture
And we did the rest the same way

b) In this section we have 200 noised images and we will need to try to restore the original images So we will use the law of large numbers to create an good image(the average of the noised images) also we can use the median for the images both of them work fine:
mean_image = np.mean(noised_images, axis=0)
or
median_image = np.median(noised_images, axis=0)

the result of the mean:

median: