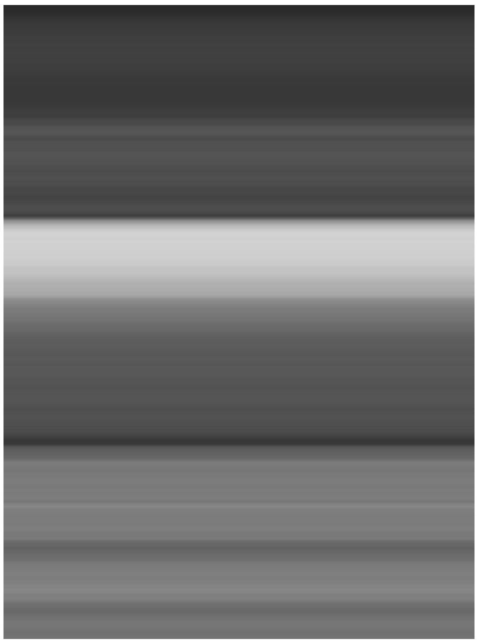
Assignment 3

Samih Warwar 324888155 – Merry Shalabi 324007202

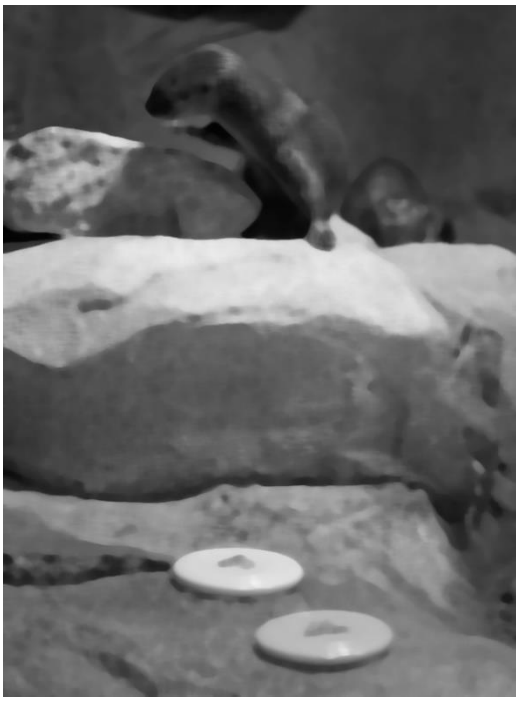
Problem1:

1. In this image we can see that in each row we have the same color that gives the average from the whole row so we used a kernel of the size of the image where all zeros and just the middle row have 1's and we got: mse=0.057.

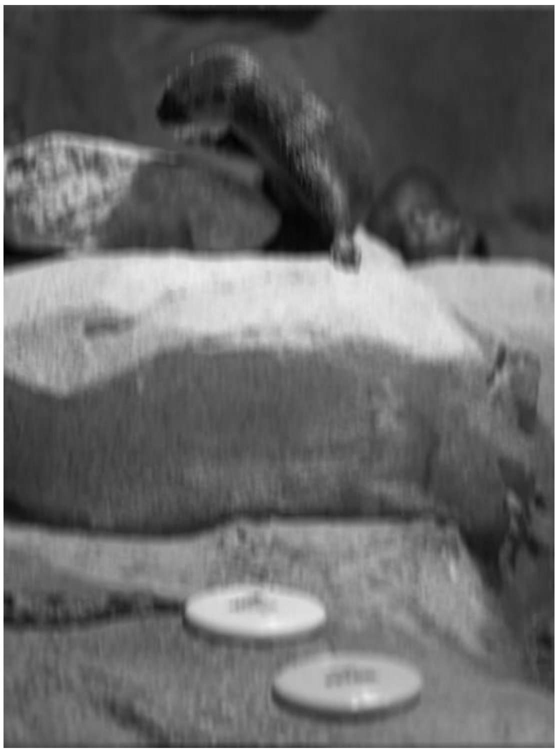


2. Here we can notice that this image has kind of soft blurring in addition to where the edges we have some kind of additional white that we took from the opposite edge, so our guess was GaussianBlur where the size of the kernel is 11 and the sigma is 10 with cyclic padding, at the end we got mse=0.218.

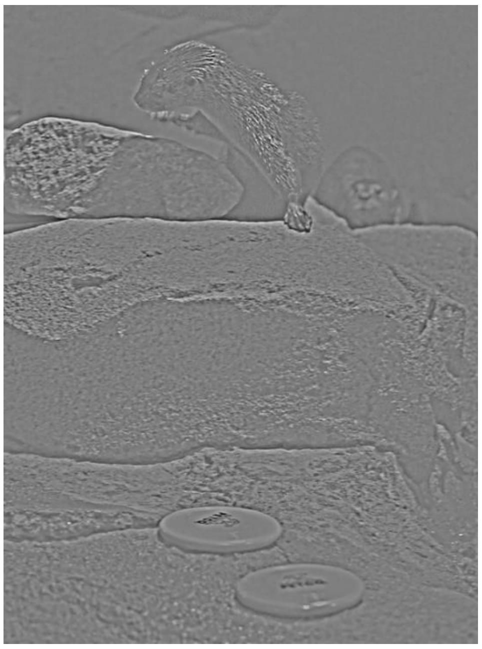
3.In this picture we can see some kind of strong blur, but we still can see the shapes clearly that’s why we went with MedianBlur with kernel size of 11 and we got mse=0.367.



4.In this picture we can see a blur more like duplication of the same image in more than place just in the y-axis, so we choose a kernel that is a column and have all 1/(kernelrows) because we wanted to protect the images average after some tries, we went with a column that have 15 row [1/15 … 1/15] and we got mse=0.315.

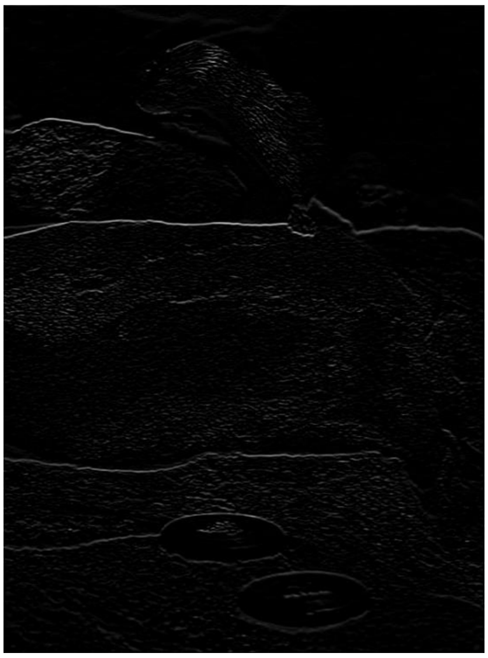


5.In the lecture we saw how can we create a sharpened image using gussian blur “Bsharp” and in the process we saw an image that has its edges popped up “high pass filter” and it looked the same one at first we did GussianBlur on the image where the size of the kernel was 17 and the sigma 4,then we removed the blurred image from the original to get the detailed one and then added some gray levels to rivel them “+127” at the end we got mse=5.665.

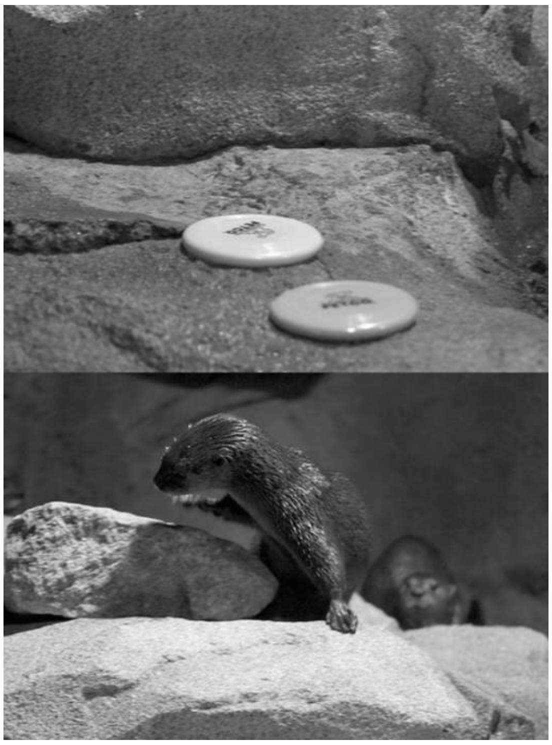


6.All black image with white edges its just Laplacian filter but we can also see that on the y-axis reveals the most, so we tried to rivel them better by choosing a kernel that look like

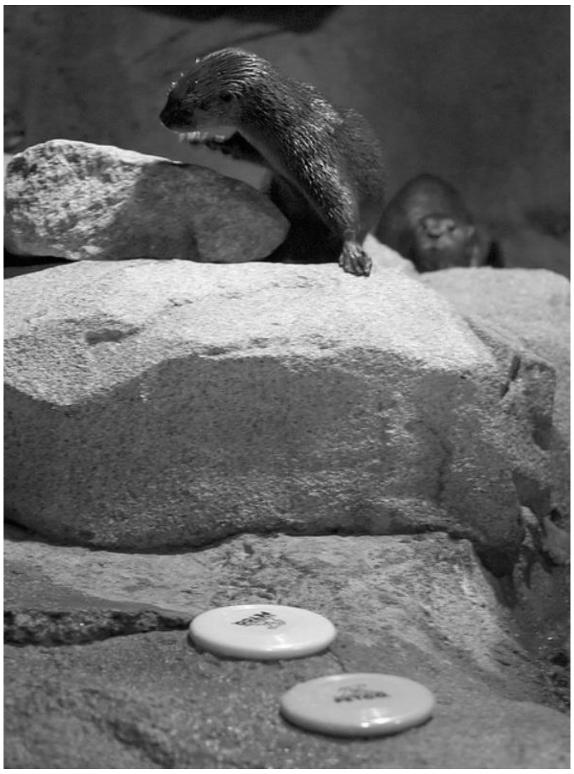
[[-0.2, -0.2, -0.2], [0, 0, 0], [0.2, 0.2, 0.2]] then we duplicate it with 1.7 to see the edges the best way at the end we got mse=5.071.



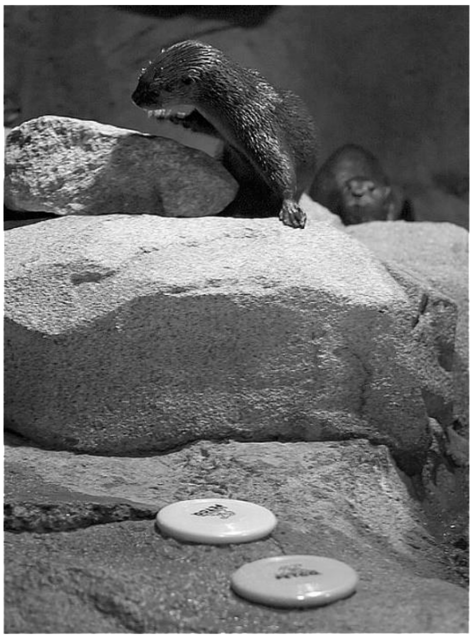
7.here the imaged is moved up (or down) the half size of the image without changing the colors so we choose a kernel to get for each pixel the pixel that appears of half image up in a cyclic way, we got mse=1.181.



8.looks exactly like the original image without blurring or additional edge revealing just turning it to grayscale so we used the ID kernel [[0,0,0],[0,1,0],[0,0,0]] and got mse=0.346.

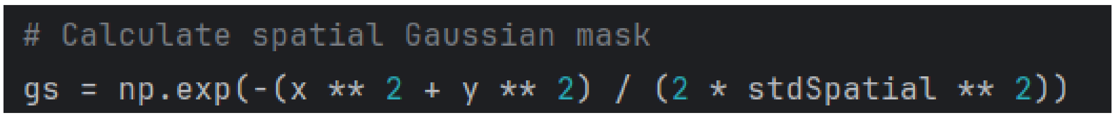


9.This one we can see the edges clearly popping and the whole image more sharp so we used a sharpening kernel where we also saved the image average, we chose [[-1,-1,-1],[-1,12,-1],[-1,-1,-1]]/4 . and got mse= 7.443.



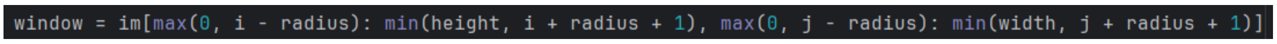
Problem 2:

a. First of all we initialize the image with 0's then we Generated indices for spatial distances (x,y) in order to use them in the following formula



```
# Calculate spatial Gaussian mask
gs = np.exp(-(x ** 2 + y ** 2) / (2 * stdSpatial ** 2))
```

then for each pixel in the image we built a window as described:



```
window = im[max(0, i - radius): min(height, i + radius + 1), max(0, j - radius): min(width, j + radius + 1)]
```

Then calculated:

the intensity difference *"intensity_diff = window - im[i, j]"*

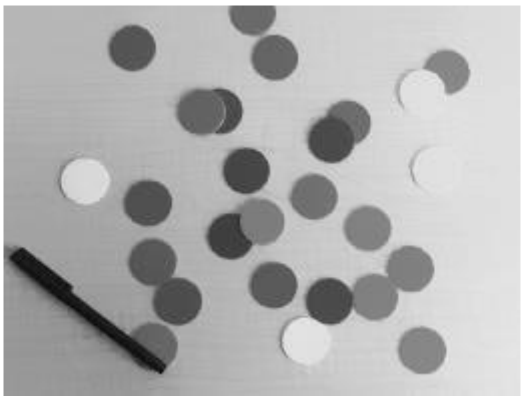
intensity Gaussian mask *"gi = np.exp(-(intensity_diff ** 2) / (2 * stdIntensity ** 2))"*

weighted sum *"weighted_sum = np.sum(gi * gs[:window.shape[0], :window.shape[1]] * window)"*

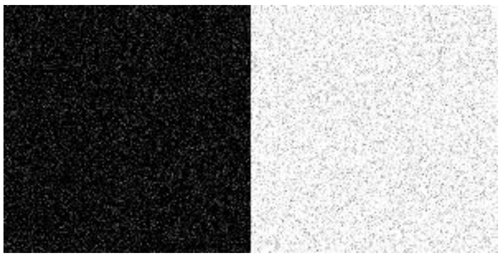
and finally the normalization factor *"normalization_factor = np.sum(gi * gs[:window.shape[0], :window.shape[1]])"*

and then updated the pixel itself by dividing the weighted_sum with the normalization_factor we already calculated.

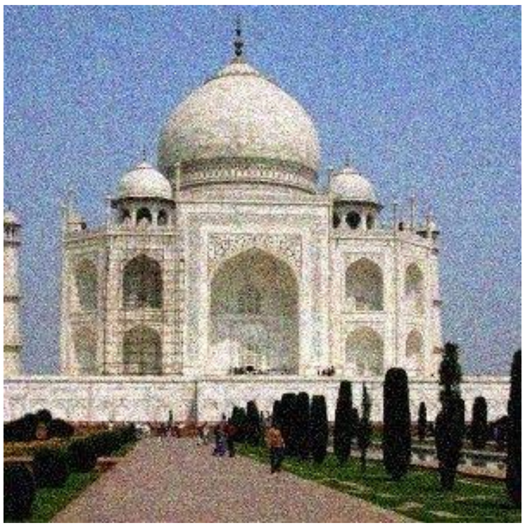
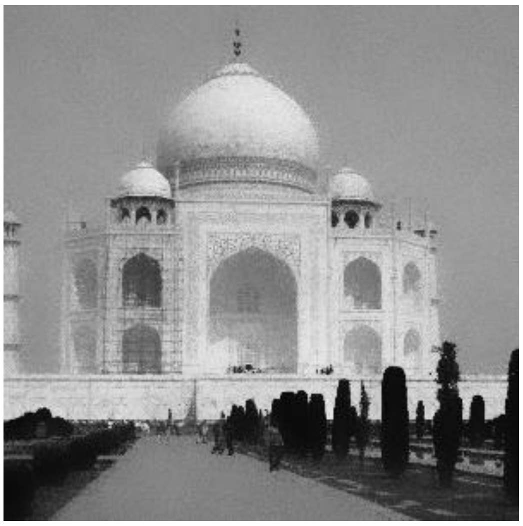
b.



for the balls image we have low resolution image, so we wanted to smoothen out the compression artifacts in the image, so we chose radius=30, stdSpatial=15 because we want to make the effect on just the near pixels and stdIntensity=10 found it the best after trying.



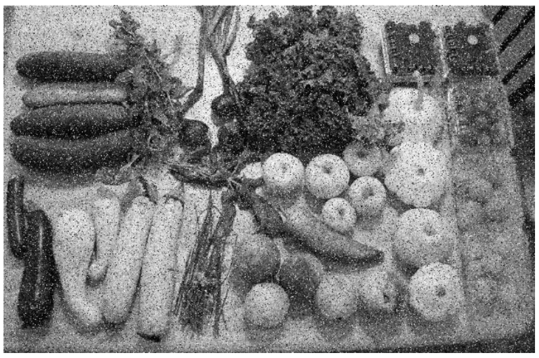
For the NoisyGrayImage we want to remove the dots from each side and because of each side has mostly the color itself (for example the right side have black pixels with white noise) we chose radius=10, stdSpatial=25 because we can get “help” from the whole section not just the pixels around but in the same time we didn’t want to mess up the line in the middle, and after trying some values we settled on stdIntensity=35.

This image has noise, so we want to smoothen out the image but at the same time we don't want to lose too many details so after trying we decided radius= 10, stdSpatial= 20 to make the smoothing on the small area and then after trying we used stdIntensity=80.

Problem3:

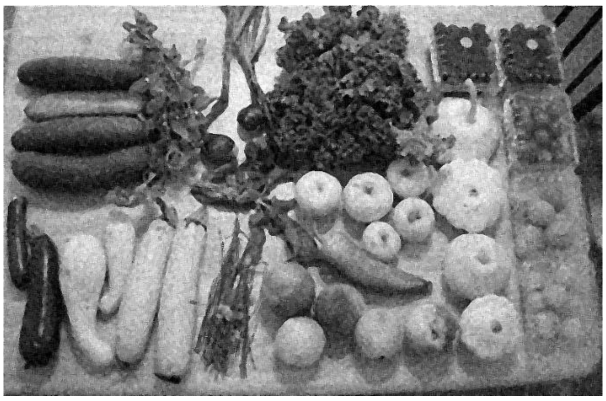
Original image:



a.To fix the image at the biggening we used medianBlur to get the noise off but as we know the edges will get smother, so we decided to add some sharpening to it.

For the medianBlur we used a kernel of size 5 after trying the 3 and 7, and for the sharpening we chose kernel = [[0,-1,0],[-1,5,-1],[0,-1,0]].

And got



b.now we can use the 200 images to get a better version of it, we started by doing the same filters we did on the first image on the whole 200 images the medianBlur and the sharpening and then for each pixel we calculated the average of the pixels from all 200 images using this line: *“cleaned_image = np.mean(deionised_image, axis=0).astype(np.uint8)”*
and got: