University of Tehran

Electrical and Computer Engineering Department

HW1 Machine Vision

Amin Fadaeinejad 810195442

Fall 2020

Abstract

In this Computer Assignment will be applying, but we have learned in real-world applications, for example, using filters in both position and frequency domain to eliminate the noise from the primary signal (currently images), changing pixel distribution of an idea, and applying Pyramid method to rescale the image, whether shrinking the image of expanding the image. We will also be using particular filters that are able to recognize the edges of the image.

1. By using the build-in Python libraries, we read the mentioned images, and by using NumPy, we were able to measures the magnitude and the phase of the images.

   (a) Figure 1 shows the input images

Figure 1: The Input images

Here are the magnitude and phase of the two given images.

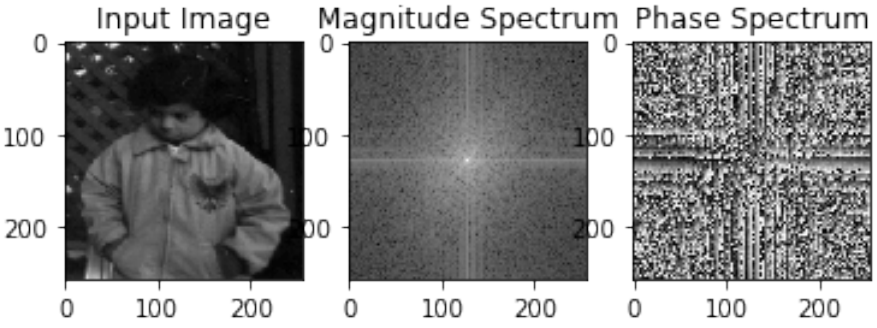

Figure 2: Magnitude and Phase of Pout

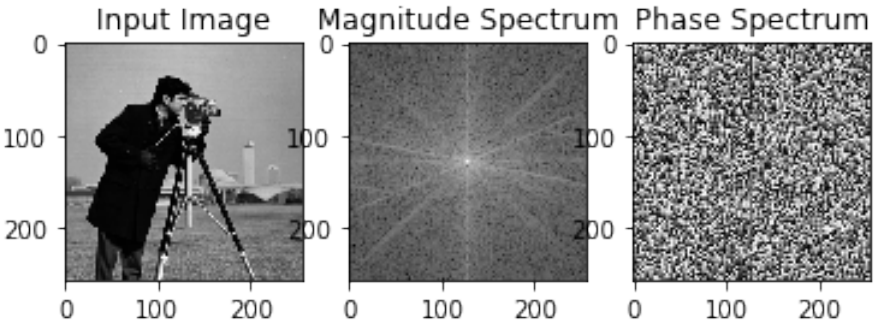

Figure 3: Magnitude and Phase of Cameraman

In the next part, we will use the magnitude of one image and phase from the other image; afterward, we will combine these two to build a new image.

$$Real\{New\ image\} = Magnitude(Image_1) \times \cos(Phase(Image_2)) \quad (1)$$

$$Image\{New\ image\} = Magnitude(Image_1) \times \sin(Phase(Image_2)) \quad (2)$$

By using equation 1 & 2 we are able to make the new image. The result will look like Figure 4 & 5.

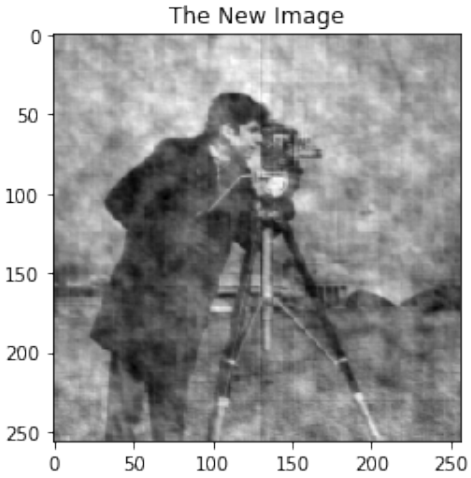

Figure 4: Magnitude Pout and Phase of Cameraman

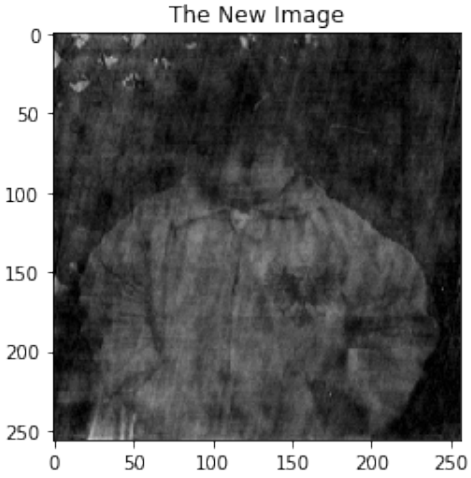

Figure 5: Magnitude Cameraman and Phase of Pout

Figure 4 shows an image with the magnitude of Pout and phase of Cameraman; Figure 5 show an image with the magnitude of Cameraman and phase of Pout. we can realize that the output is more dependent on the phase compared to the magnitude.

(b) In the next part we are going to use the magnitude of the previous images and apply a constant value for the phase.

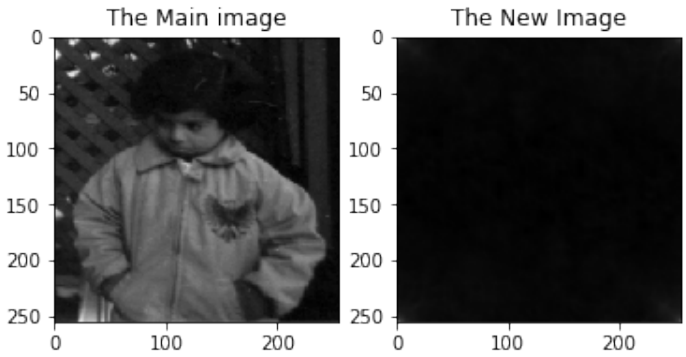

Figure 6: Magnitude Pout and Phase of constant

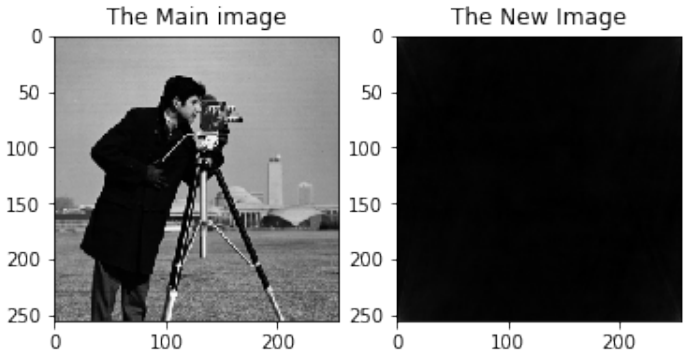

Figure 7: Magnitude Cameraman and Phase of constant

Figures 6 and 7 shows that only the magnitude of the frequency domain can't help to recover the image, and the phase of an image is much more important. The reason is that the magnitude just shows the magnitude of the sin and cos of the image but will not give us the phase and the direction of the image. The phase shows where the sin and cos function will reach each other and will make a line on the output.
Therefore phase plays a much more effective roll compared to the magnitude.

2. We have an image which it's pixels aren't uniformly distributed, and the goal of this section is uniformly distributing the pixels. In the beginning, we will plot the histogram of the current image.

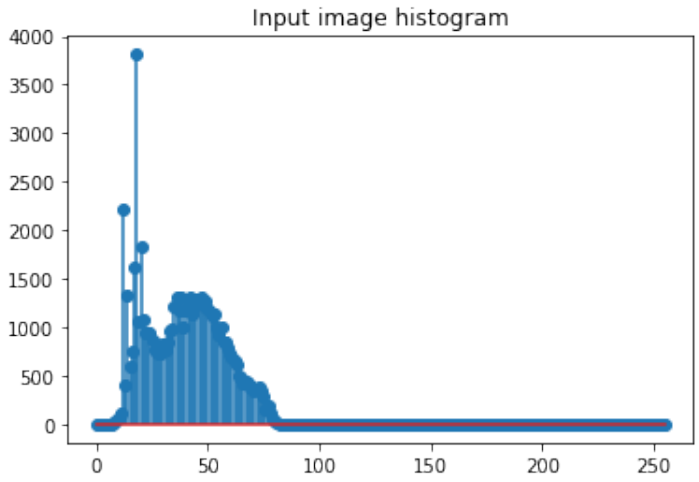

Figure 8: Input image histogram

After that we are going to use the CDF of the pixel's distribution to estimate to calculate the proper new value of each pixel. The way we are going to use Histogram Equalization in this question is mentioned below:

(a) We need to multiply the CDF in the possible number of outputs - 1, and that number for us is 255.

(b) In the next part, we need to check for every particular input what is the new figure (after multiplying in 255).

(c) After that, we are going to round up the new number (because the number might have fraction).

(d) In the end, we will use all the data we have, apply the method, and plot the new histogram of the data.

(e) In the end, we can remake our image with the new data we have.

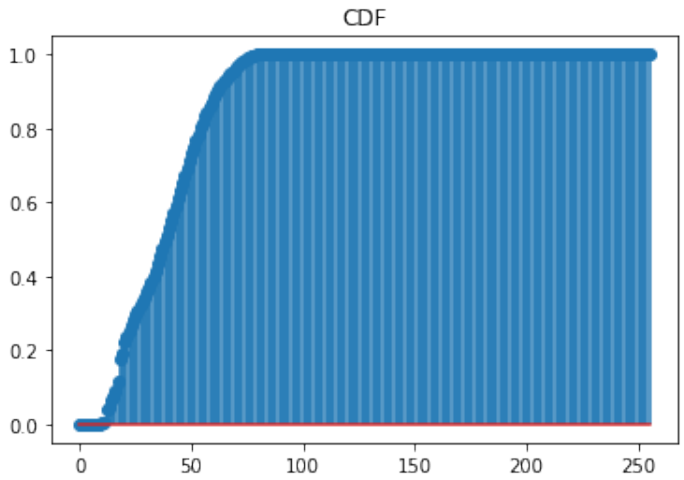

Figure 9: CDF of the pixels

Figure 9 shows the CDF of the image, as you can see most of the pixels have low value.

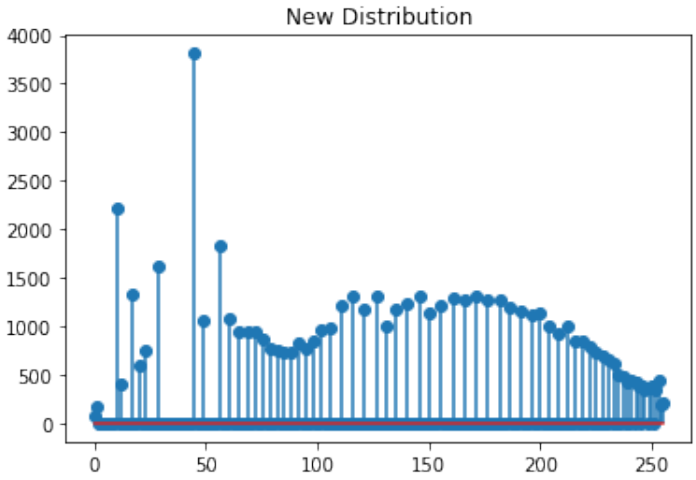

Figure 10: The new histogram

After applying the method the new histogram will look like figure 10. As you can see the distribution is quite uniform compared to the first image.

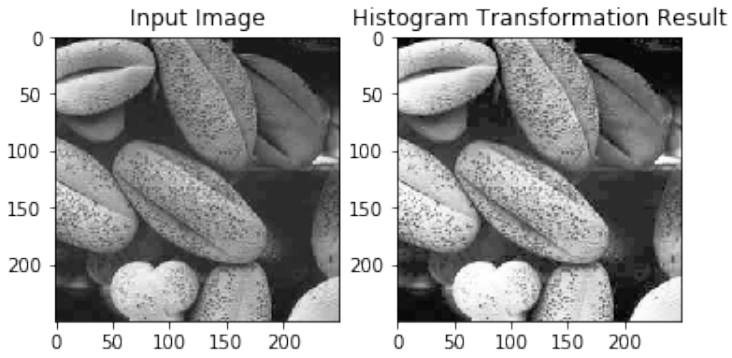

Figure 11: Comparing two images

Figure 11 shows the image in the beginning and image in the output. As you see the new image seems quite brighter compared to the image in the first place.
Conclusion: By applying the Histogram Equalization method to an image, we will receive an image in which pixels are uniformly distributed. This method will help us change the image that is much easier to understand than other images that are too bright or too dark. We can also use this kind of method in order to change the distribution to any kind of distribution we want just by following the principles.

3. In this question I we used three different methods in order to eliminate the noise here are the following ways:

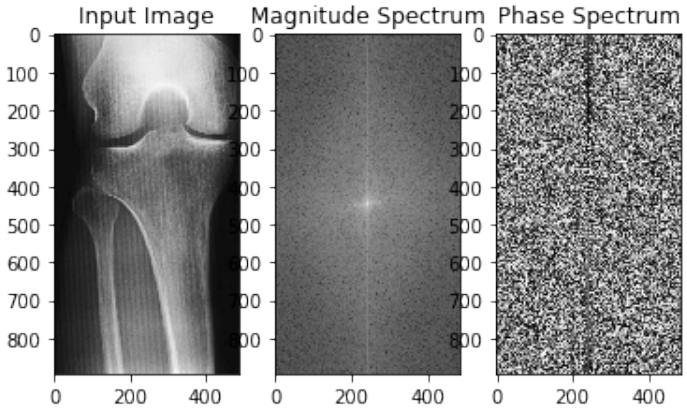

Figure 12: Main image and the magnitude, phase

(a) The first one is using a 7×7 average kernel which we are going to use in position domain and convolve it with the image.

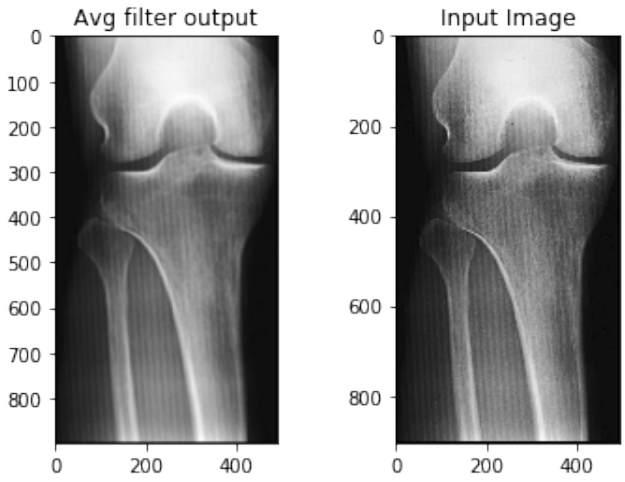

Figure 13: Main image and the magnitude, phase

As you can see in figure 13, the avg filter had an effect on the image. Some of the main image lines were cleaned after this filter, but still, the noise wasn't cleaned thoroughly.

(b) The second method was applying a Gaussian filter to the image (In frequency domain)

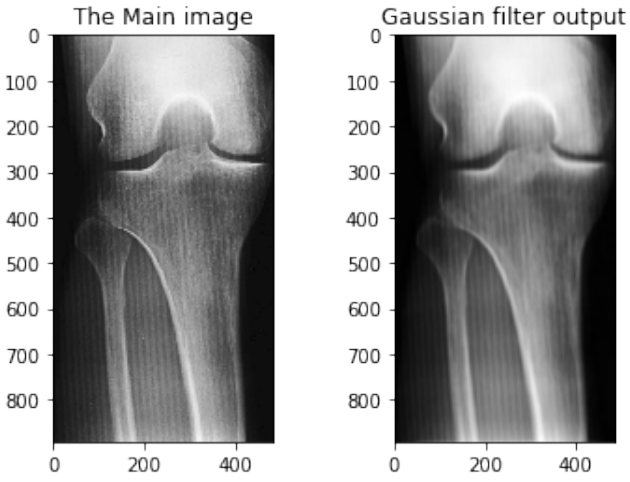

Figure 14: Main image and the magnitude, phase

As you can see in figure 14, the gaussian filter had effect on the image and some of the lines where cleaned, the point is that this filter had a better effect

compared to the previous filter.

(c) In the last method we viewed the image in frequency domain and saw some bright spots on it.

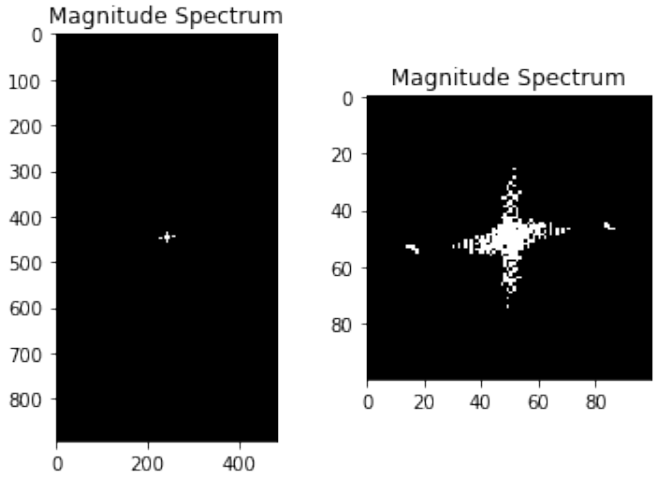

Figure 15: Main image and the magnitude, phase

As you can see in figure 15, around the middle of the frequency domain, there are a few spots where the magnitude is exceptionally high, causing the noise on the bone image.

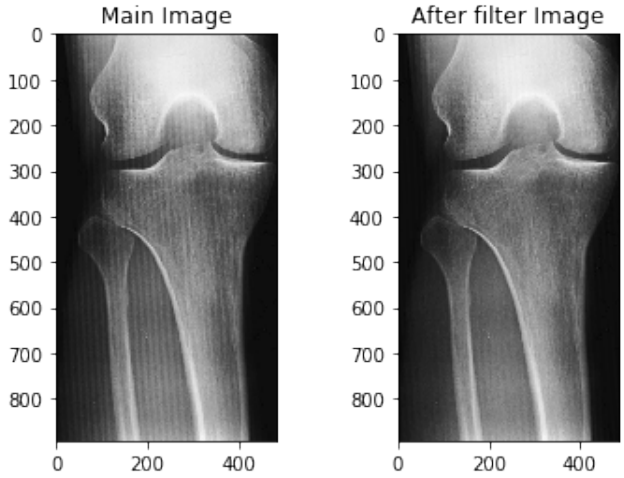

Figure 16: Main image and the magnitude, phase

Using a filter that gets rid of those spots, we can recover the image, and the result will look like figure 16. As you can see, this was the best method of them all to eliminate the noise.

4. In this part we are going to compare two different filters, which one is Ideal and the other on is a Gaussian.

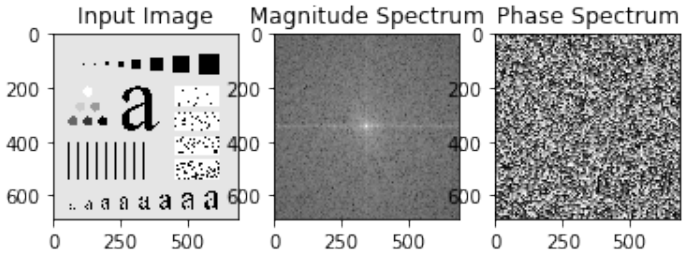

Figure 17: Main image and the magnitude, phase

We can see the image and it's magnitude and phase in the frequency domain in figure 17. $F(u,v) = H(u,v) \times G(u,v) \& D_0 = 35$

(a) Applying Gaussian filter:

$$(position\ domain)\ G_\sigma = \frac{1}{2\pi^2} e^{-\frac{x^2+y^2}{2\sigma^2}} \tag{3}$$

$$(frequency\ domain)\ H(u,v) = e^{-\frac{D(u,v)^2}{2{D_0}^2}} \tag{4}$$

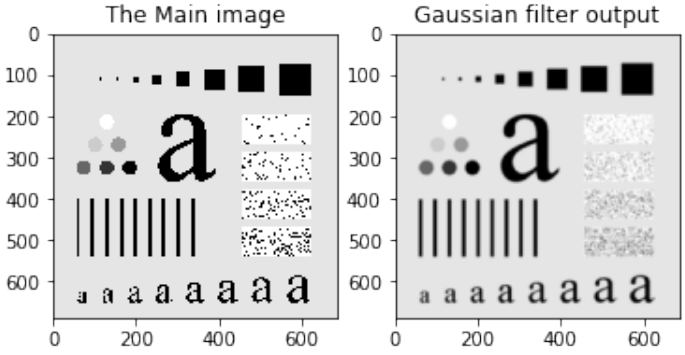

Figure 18: Main image and the magnitude, phase

(b) Applying Ideal filter:

$$if\ D(u,v) \leq D_0 \rightarrow H(u,v) = 1 \tag{5}$$
$$if\ D(u,v) > D_0 \rightarrow H(u,v) = 0 \tag{6}$$

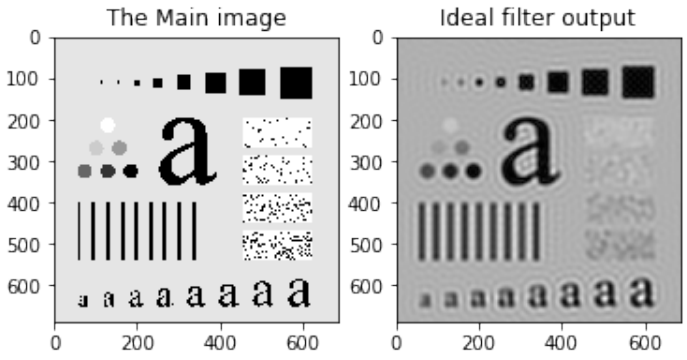

Figure 19: Main image and the magnitude, phase

5. We assume that the mask is padded with zero which means the elements that are not shown in the question are zero.

$$F(u,v) = \sum_{x=-\infty}^{\infty} \sum_{y=-\infty}^{\infty} f(x,y)e^{-j(xu+yv)} \rightarrow F(u,0) = ? \tag{7}$$

We are going to calculate $F(u,0)$, we assume that $\forall x||y \notin \{-1,0,1\}\ f(x,y) = 0$ therefore:

$$F(u,0) = \sum_{x=-1}^{1} \sum_{y=-1}^{1} f(x,y)e^{-j(xu+0)} = f(-1,0) \times e^{ju} + (f(0,-1) + f(0,0) + f(0,1)) \times e^{0} + f(1,0) \times e^{-ju}$$

$$= 2 - e^{ju} - e^{-ju} = 2(1 - \cos u) \rightarrow F(u,0) = 2(1 - \cos u)$$

So as we can see, the frequency domain of the Laplacian mask has no imaginary past, and it's all real. Figure 20 represents the magnitude of the filter.
Discrete Laplace operator is often used in image processing, e.g., in edge detection and motion estimation applications. The discrete Laplacian is defined as the sum of the second derivatives Laplace operator Coordinate expressions and calculated as the sum of differences over the nearest neighbors of the central pixel. Since derivative filters are often sensitive to noise in an image, the Laplace operator is often preceded by a smoothing filter (such as a Gaussian filter) in order to remove the noise before

calculating the derivative. The smoothing filter and Laplace filter are often combined into a single filter.

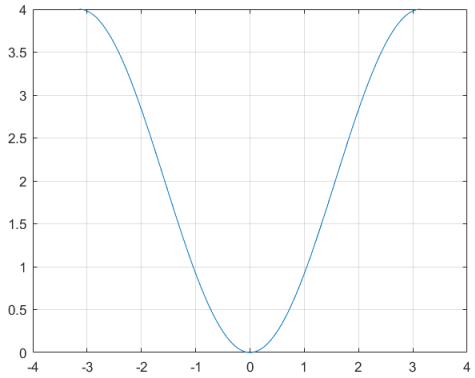

Figure 20: Magnitude of the mask in frequency domain

Because these kernels are approximating a second derivative measurement on the image, they are very sensitive to noise. To counter this, the image is often Gaussian smoothed before applying the Laplacian filter. This pre-processing step reduces the high-frequency noise components prior to the differentiation step.
Since the convolution operation is associative, we can convolve the Gaussian smoothing filter with the Laplacian filter first and then convolve this hybrid filter with the image to achieve the required result. Doing things this way has two advantages:

(a) Since both the Gaussian and the Laplacian kernels are usually much smaller than the image, this method usually requires far fewer arithmetic operations.

(b) The log (‘Laplacian of Gaussian’) kernel can be precalculated in advance so only one convolution needs to be performed at run-time on the image.

Figure 20 can show that for this filter will let the edges pass the filter while the other section of the image will be filtered.

6. In this part we are going to use the Laplacian Kernel for edge detection of the given image

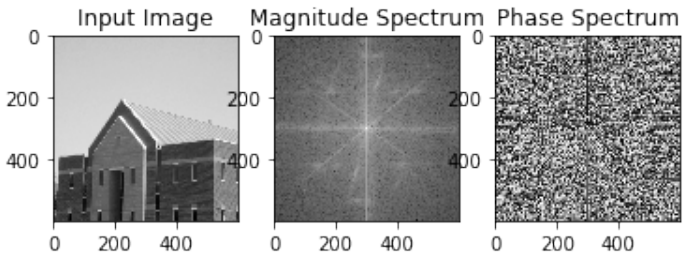

Figure 21: Main image and the magnitude, phase

Figure 21 represents the image magnitude and phase in the frequency domain. In the next part, we are going to use the Laplacian filter in position and frequency domain in order to see if this filter is able to detect the edges of the image.

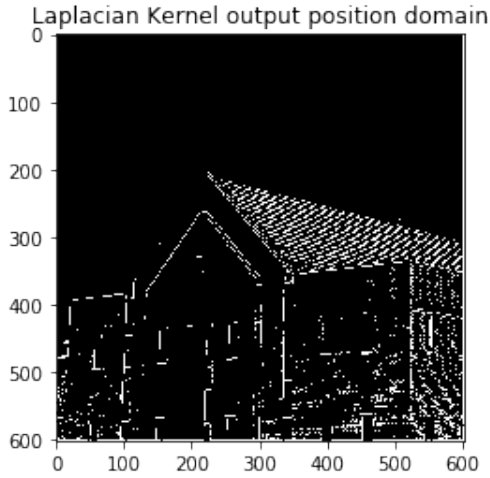

Figure 22: Main image and the magnitude, phase

Figure 22 displays the result of the edge detection in position domain, by convolving laplacian filter with the image and afterword applying a threshold in order to detect the value of pixels which are greater than a particular value.

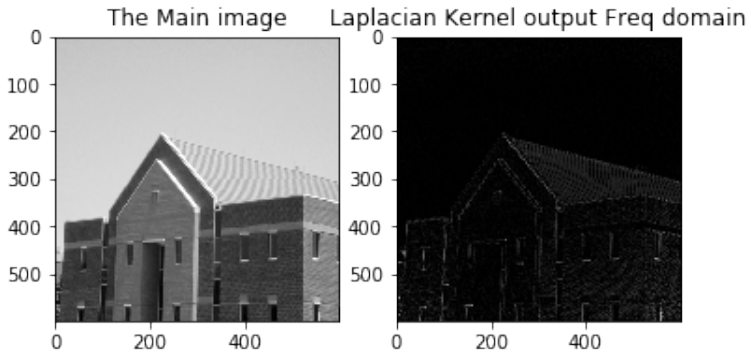

Figure 23: Main image and the magnitude, phase

Figure 23 represents the image after using a Laplacian filter in the frequency domain. We need to zero pad the filter at the beginning in order to be able to multiply two matrices in an element-wise mode.
As we can compare the two methods, we can see that both of them have a similar effect on the output because we just used the frequency domain of the same filter. To be more idealistic, we must get the exact same result because, as we all know, convolving is the position domain is similar to multiplying in the frequency domain.

7. In this part, we will use different things we have learned till now to filter a three channeled RGB image.
In this first part, we need to divide the RGB image into three different color channels.

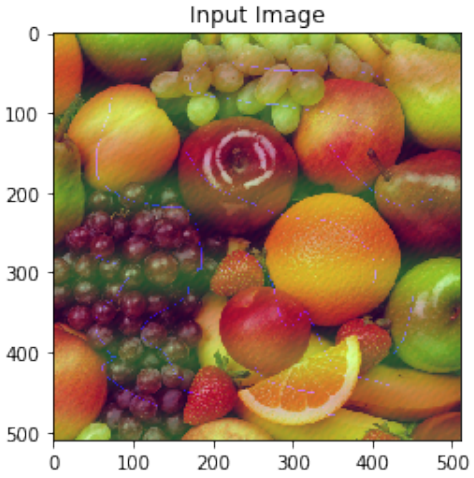

Figure 24: The Noisy image

Figure 24 shows the noisy image, and the objective of this question is to remove the noise from every single channel of this image.

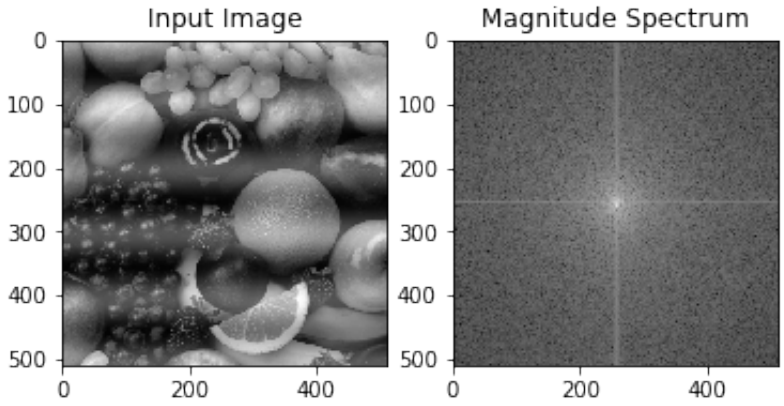

Figure 25: The G channel image and frequency domain

Figure 25 shows the image for the green channel and it's frequency domain, as you can see the image consists of the wave noise.

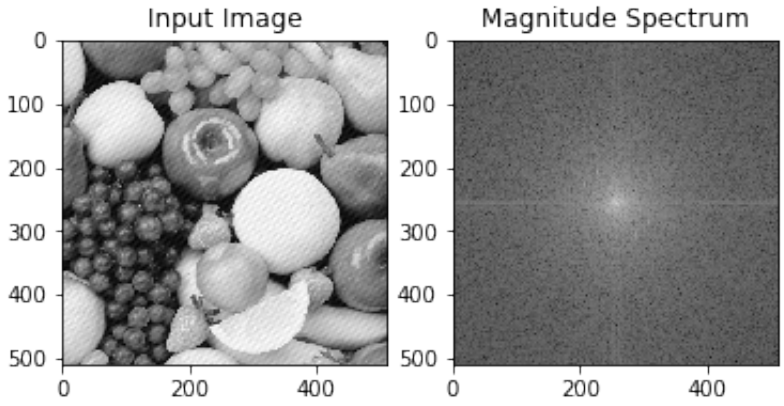

Figure 26: The R channel image and frequency domain

Figure 26 shows the Red channel's image and its frequency domain; as you can see, there is a wave noise which it's direction is from the top left to the downright.

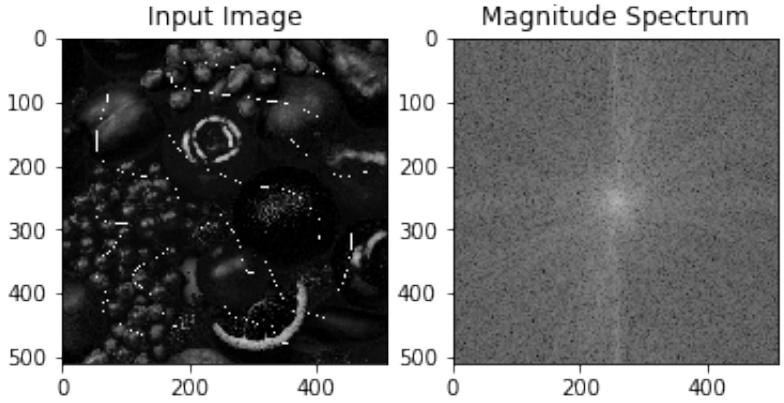

Figure 27: The B channel image and frequency domain

Figure 27 shows the image for the Blue channel and its frequency domain; as you can see, there is a line on the image that doesn't have a particular pattern. Therefore, we are going to use a different method in order to get rid of that line.
Now we are going to show how we get eliminate the noise from every filter in this three channels.

(a) Green Channel: In this part, we needed to have a better look at the frequency domain of the channel; after magnifying the image and comparing the value near the center of the frequency domain, we realized that in the frequency domain, there exist some particular spots that have an extremely high value and we need to get rid of them in order to eliminate the noise.

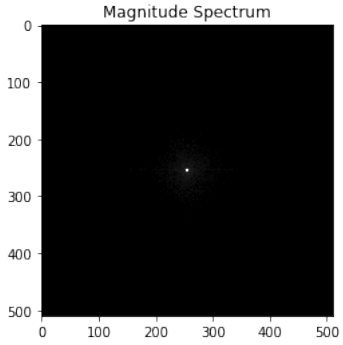

Figure 28: The B channel image and frequency domain

Figure 28 shows the frequency domain of the image. In this view, we are not able to see the details in the image, so we need to zoom on the image to have a

better look.

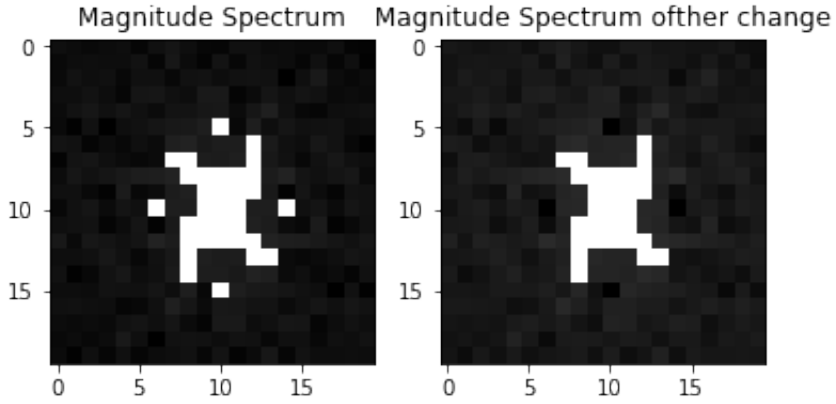

Figure 29: The B channel image and frequency domain

Figure 29 shows the center of the frequency domain, and as you can see, there are four spots that have a high value, and they are the cause of the noise; out objective is to use a mask a get rid of those spots, and in the end, recover the main channel.

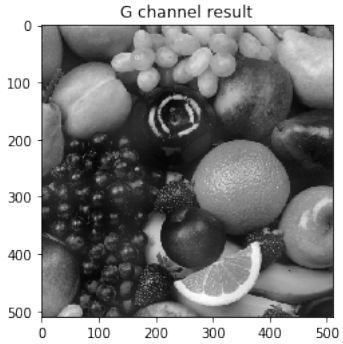

Figure 30: The B channel image and frequency domain

Figure 30 shows the new image of the Green channel as you can see the noise have vanished.

(b) Red Channel: In this part we needed to have a better look on the frequency domain of the channel, after magnifying the image and comparing the value of near the center of the frequency domain we realized that in the frequency domain there exists some particular spots that have a extremely high value and we need to get rid of them in order to eliminate the noise.

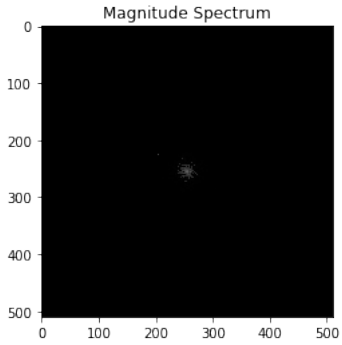

Figure 31: The B channel image and frequency domain

Figure 31 shows the frequency domain of the image, in this view we are not able to see the details in the image so we need to zoom on the image to have a better look.

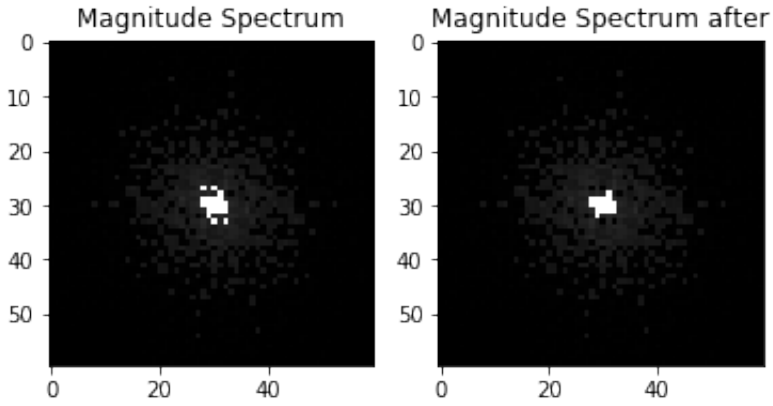

Figure 32: The B channel image and frequency domain

Figure 32 shows the center of the frequency domain, and as you can see, there are two spots that have a high value, and they are the cause of the noise; out objective is to use a mask a get rid of those spots, and in the end, recover the main channel.

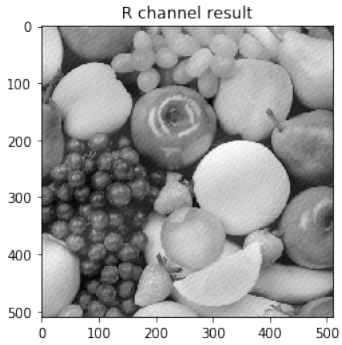

Figure 33: The B channel image and frequency domain

Figure 33 shows the new image of the Green channel as you can see the noise have vanished.

(c) Blue Channel: We can notice from the image that the lines have a very high value. In this case, we can measure the value in every pixel in the image, and if the value of the particular pixel is higher than a certain threshold, then we will replace the value of that pixel with the average pixel value of the entire image. In the end, our threshold was 200, and Figure 28 is the result of this method for eliminating the line.

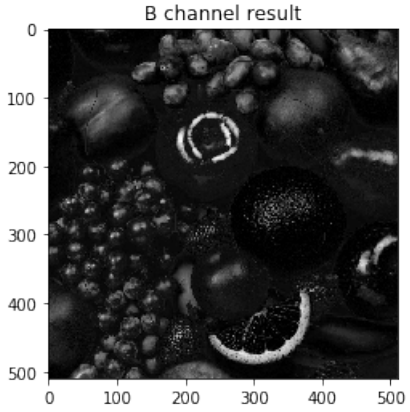

Figure 34: New Channel B

In the end by combining this three channels we will get a RGB image that will look like figure 35.

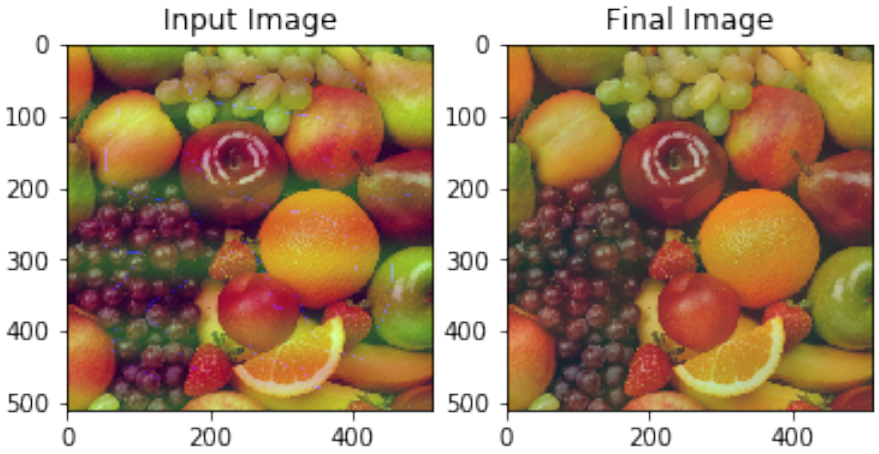

Figure 35: Before and After operation

As you can see, by using the mentioned methods, we were able to eliminate the noise from the image even though if we just used a normal filter (Ideal or Gaussian), we weren't able to reach this result.

8. In this part we are given two images, which one has noise and the other hasn't.

   (a) Figure 36 will represent the noisy image and it's frequency domain.

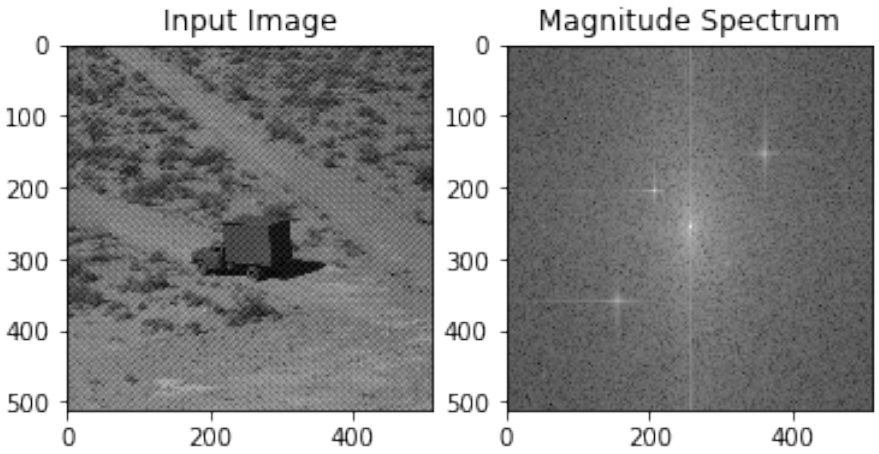

Figure 36: Image and frequency domain

After having a search, we realized that the type of noise that we are dealing with is a periodic noise. The reason that we guess that the current image has

the periodic noise is that we can clearly see a periodic wave on the image and in other point there are four bright spots on the frequency domain of the image, which represents that there a number of periodic elements in the image and the position where those bright spots are located will show the direction and the frequency of those periodic signals. Here is a brief about this noise:

Periodic noise

A common source of periodic noise in an image is from electrical or electromechanical interference during the image capturing process. An image affected by periodic noise will look like a repeating pattern has been added on top of the original image. In the frequency domain, this type of noise can be seen as discrete spikes. A significant reduction of this noise can be achieved by applying notch filters in the frequency domain. The following images illustrate an image affected by periodic noise and the result of reducing the noise using frequency domain filtering. Note that the filtered image still has some noise on the borders. Further filtering could reduce this border noise. However, it may also reduce some of the fine details in the image. The trade-off between noise reduction and preserving fine details is application-specific. For example, if the castle's fine details are not considered important, low pass filtering could be an appropriate option. If the fine details of the castle are considered necessary, a viable solution may be to crop off the border of the image entirely.

(b) As we can see from figure 37, the image has four bright spots, which represent to frequency domain of the noise on the signal. In this case the objective of this part is to filter those bright spots and recover the previous image.

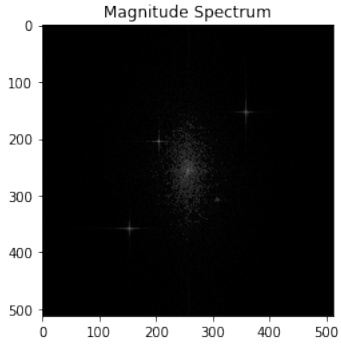

Figure 37: Magnified frequency domain

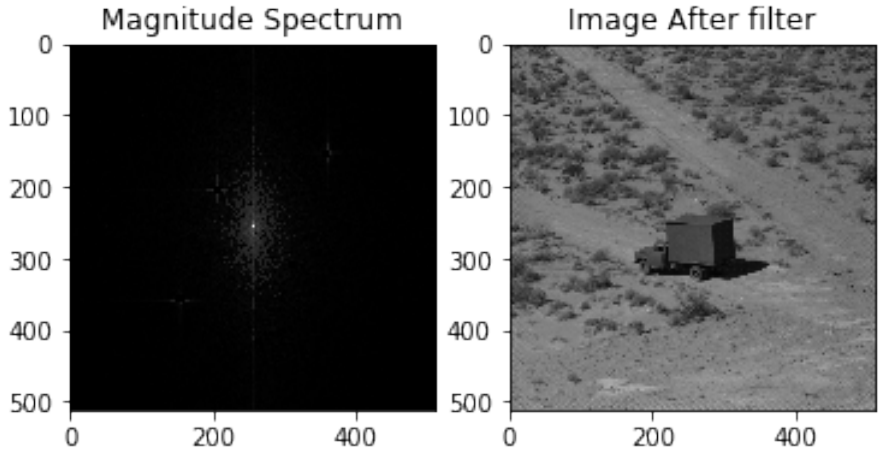

Figure 38: Image after applying filter

In figure 38 we can see the we applied a filter that will to reduce the effect of the four spots that we have found in the previous part, and by applying that in recovering the image we will get figure 38.
As you can see the image is mush clearer compared to the first image.
In this part we are going to use a filter in frequency domain, because implementing a filter with this frequency domain in the position domain is very complicated and if we just use a normal filter we will end up erasing important information from the signal, so the best thing that we can do is to use a particular filter in frequency domain.

(c) In the last part we need to measure the SNR and PSNR parameters of the signal before and after applying the method from part B.

$$SNR_{dB} = 10\log_{10}(\frac{P_{signal}}{P_{Noise}}) \tag{8}$$

$$MSE = \frac{1}{MN}\sum_{i=0}^{m-1}\sum_{j=0}^{n-1}(I(i,j) - K(i,j))^2$$

$$PSNR = 10\log_{10}(\frac{MAX_I^2}{MSE}) = 20\log_{10}(MAX_I) - 10\log_{10}(MSE) \tag{9}$$

By using equations 8 & 9 we can calculate this two parameters before and after applying the filter to the noisy image.

Method	Value
SNR before filter	17.330300
SNR after filter	30.54198
PSNR before filter	24.59542
PSNR after filter	37.80710

As you can see the result shows that the SNR increases when we apply our filter which shows that the filter had significant effect on reducing the noise, and also the PSNR is increasing which means the MSE parameter is decreasing which overall means the mean square error had decreased so in this can this also shows that our filter had a good effect of the noisy image

9. this is Q9 In this part we are going to lean about how we can use the Pyramid method in order to shrink the dollar image. We going to divide this image by two for two times. In every step in this division we need to filter the image at first in order to get rid of the noise and in this case we used the guassian filter and in every step the σ of the filter need to change by $\frac{\sqrt{3}}{2}$ every time so we need to implement this at the beginning

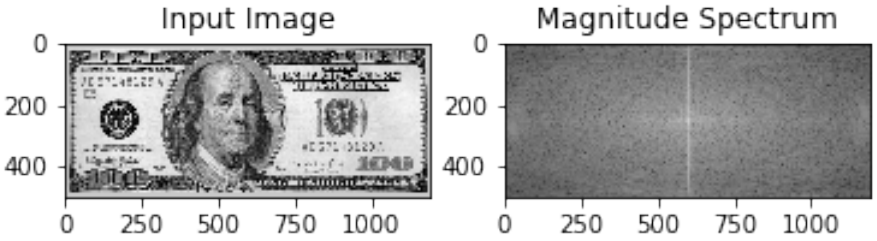

Figure 39: Image after applying filter

Figure 39 shows the image and it's frequency domain.

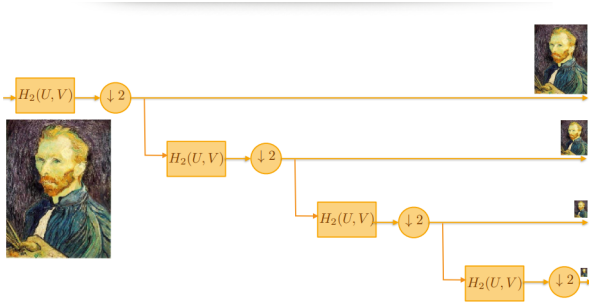

Figure 40: Pyramid algorithm

Figure 40 shows the pyramid algorithm that we are going to implement in the part

Figure 41: Image after applying filter

Figure 41 shows the image after applying the guassin filter.

Figure 42: Image after applying filter

Figure 42 shows the image after sub sampling the image by division two.

Figure 43: Image after applying filter

Figure 43 shows the image after applying the guassin filter to the shrink image.

Figure 44: Image after applying filter

Figure 44 shows the image after sub sampling the image by division two which means it's 4 times smaller than the main image.

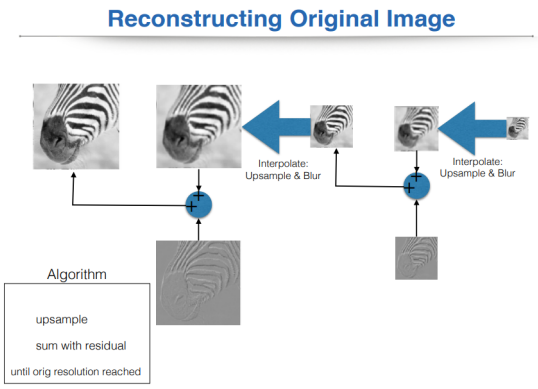

Figure 45: Reconstruct algorithm

Figure 45 shows the algorithm for remaking the image.

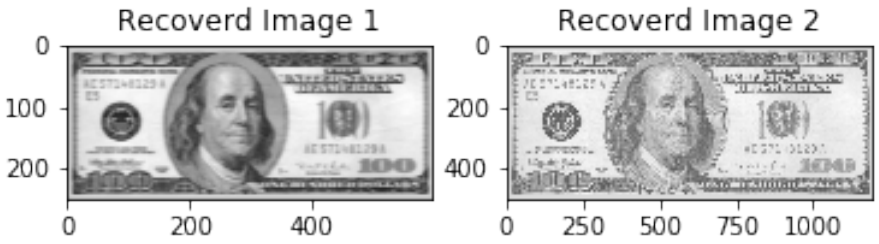

Figure 46: Reconstruct algorithm

Figure 46 shows the two dollar images that we have recovered from the shrink image. The Recovered image 1 half the size of the main dollar and Recovered 2 is the same size of the main dollar.
As the conclusion we were able to recover the shrink image.