CMPE362 HW3

Frequency Domain Processing

Eren Kılıç

Department of Computer Engineering

TED University

I- INTRODUCTION

In this assignment, we are asked to use and test given input images by applying frequency domain processing techniques and compare them with different parameters. In the first part, we required to smoothing the image using frequency domain filters by applying the Ideal low pass filter, Butterworth low pass filter and gaussian low pass filter. Then in the second part, we should have repeated similar processes as the first one since we needed to use frequency-domain for image sharpening. Yet, this time, I applied three types of high pass filters for sharpening. Lastly, for the third part, we are asked to use notch filters to suppress Moire patterns from input images.

A detailed explanation of the steps is given below.

II- STEPS FOLLOWED

PART 1: Image Smoothing Using FDF

a) Ideal Low Pass Filter

The main purpose of this part was related to applying Ideal low pass filter with different cutoff frequencies. To complete this part, firt I read the input with imread function then I calculated input's row and column sizes for the next operations.

Figure 1: *Input image (panda.png)*

Afterward, I created a filter H with full of zeros (black) which has the same size with my input image. Then by iterating over image (two nested loops) I changed some pixels value to 1 (made them white). I used the formula below to create low pass filter.

The transfer function for the ideal low pass filter can be given as:

$$H(u,v) = \begin{cases} 1 \text{ if } D(u,v) \le D_0 \\ 0 \text{ if } D(u,v) > D_0 \end{cases}$$

where $D(u,v)$ is given as:

$$D(u,v) = [(u - M/2)^2 + (v - N/2)^2]^{1/2}$$

Figure 2: *Ideal Low pass formula*

This formula helped to create a circle in the middle of the filter. After the execution of the loops, H became as follow.

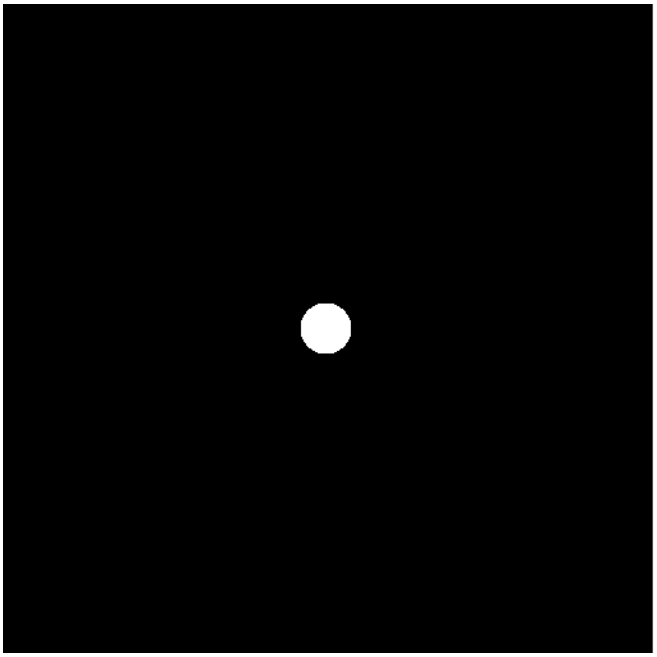


Figure 3: *H (Ideal low pass filter, cutoff = 20)*

Figure 4: Ideal low pass *output image (cutoff = 20)*

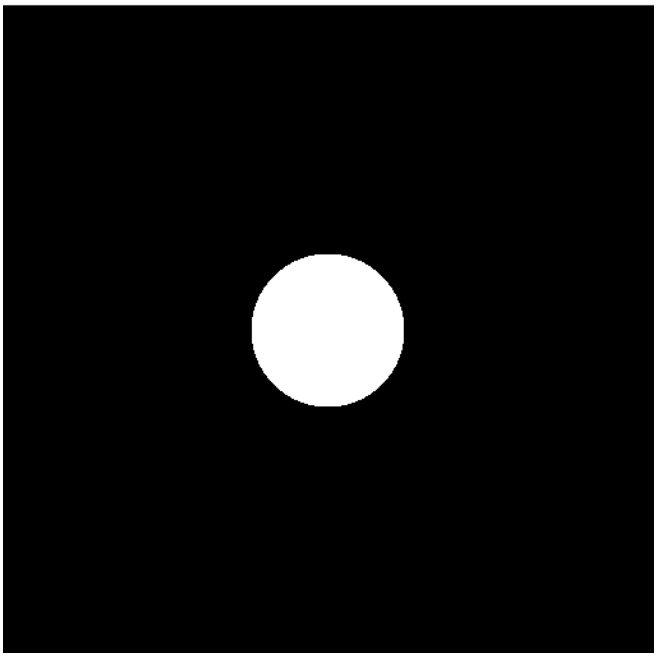


Figure 4: *H (Ideal low pass filter, cutoff = 60)*

Figure 5: Ideal low pass o*utput image (cutoff = 60)*

By checking figure 3 and figure 4, we can see that the size of the circle in the middle is related to cutoff frequency value. The size of the circle is directly proportional to the value of the cutoff value.

After that, I put zero freq in the upper left corner. Then I transformed my image with fft2 MATLAB's built-in function; I multiplied filter by Fourier image to apply the filter then did inverse FFT. I got following outputs with two different cutoff frequency values.

If we compare, figure 4 with figure 5 we see that, figure 4 is more smoothed than figure 5 because of the cutoff frequency values. So, we can conclude that, if the cutoff frequency of the filter is low, the output image becomes quite blurred and not very clear. However, if the cutoff frequency is higher, the output image becomes less blurred and more clearer.

Beside that, ringing can be observed on ideal low pass filtered images. The ringing effect increases when the cutoff value is low.

b) Butterworth Low Pass Filter

For this part, I followed the same steps as I did on the previous part. The only difference was on the use of the formula since Butterworth formula is different. I arranged the filter image H by following this formula:

The transfer function of a Butterworth lowpass filter of order n with cutoff frequency at distance D_0 from the origin is defined as:

$$H(u,v) = \frac{1}{1+[D(u,v)/D_0]^{2n}}$$

Figure 6: *Butterworth Low pass formula*

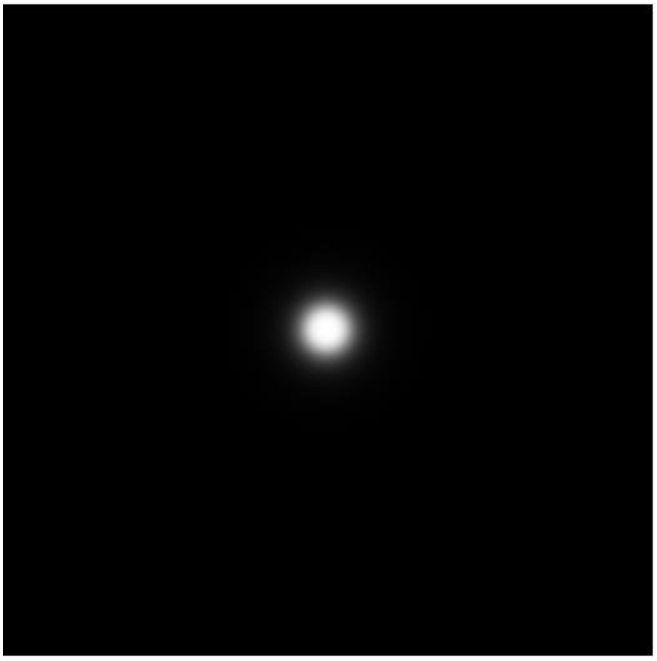


Figure 7: *H (Butterworth low pass filter, cutoff = 20)*

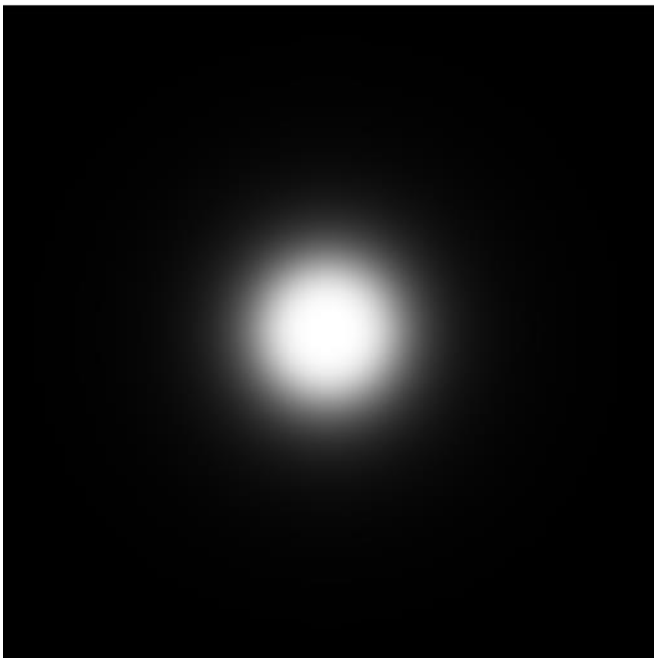


Figure 8: *H (Butterworth low pass filter, cutoff = 60)*

Figure 9: *Butterworth low pass filtered output image (cutoff = 20, order = 2)*

Figure 10: *Butterworth low pass filtered output image (cutoff = 60, order = 2)*

If we check figure 7 and figure 8, we notice that they are different compared to figure 3 and figure 4 because Butterworth filter keeps some high frequency information. Also, images smoothed finer than ideal low pass filter as well as there is no ringing here because the edges are soft in filter. Again, we see that greater cutoff means less smooth and more detailed image and less cutoff means more blurred and less detailed image.

c) Gaussian Low Pass Filter

To complete the first part, I followed the same steps as I did on the previous parts. The only difference was again on the use of the formula since Gaussian formula is different. I arranged the filter image H by following this formula:

The transfer function of a Gaussian lowpass filter is defined as:

$$H(u,v) = e^{-D^2(u,v)/2D_0^2}$$

Figure 11: *Gaussian Low pass formula*

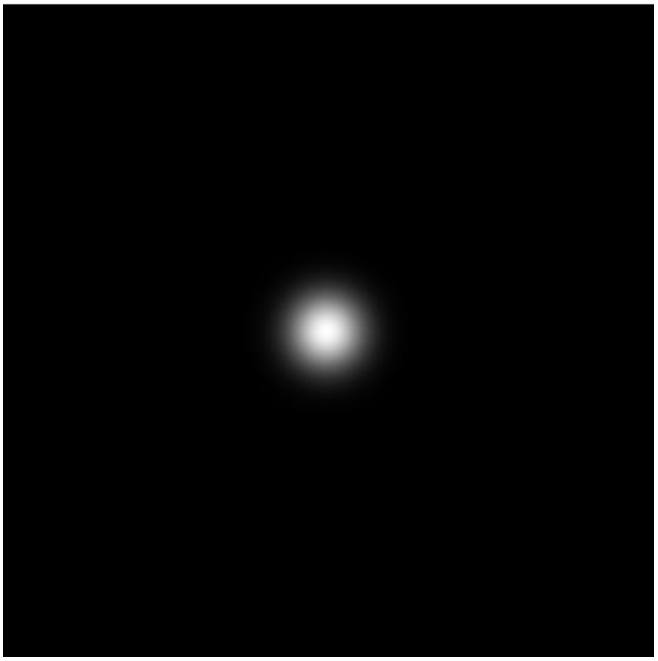


Figure 12: *H (Gaussian low pass filter, σ = 20)*

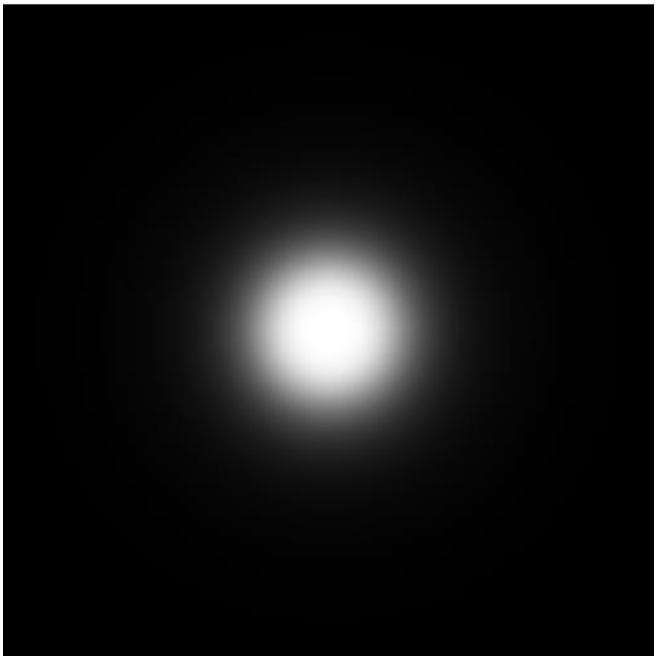


Figure 13: *H (Gaussian low pass filter, σ = 60)*

Figure 14: *Gaussian low pass filtered output image (σ = 20)*

Figure 15: *Gaussian low pass filtered output image (σ = 60)*

If we check figure 12 and figure 13, again we notice that they are different compared to figure 3 and figure 4 because Gaussian filter keeps some high frequency information like Butterworth filter. Also, images smoothed better than ideal low pass filter. Additionally we realize that, if we increase sigma value, image becomes more detailed and less blurred. This filter is quite alike to Butterworth filter. No ringing since edges are soft.

PART 2: Image Sharpening Using FDF

High pass filters only pass the high frequencies, drop the low ones. High pass frequences are precisely the reverse of low pass filters, so:

$$H_{hp}(u, v) = 1 - H_{lp}(u, v)$$

a) Ideal High Pass Filter

This part of the assignment was relevant to applying Ideal high pass filter with different cutoff frequencies. To complete this part, I followed similar procedures as before. I read the input with imread function then I calculated input's row and column sizes for the next operations.

Figure 16: *Input image (panda.png)*

Like part 1 of the homework, I created a filter H with full of zeros (black) which has the same size with my input image. Since the high pass filter is the reverse of the low pass filters, I repeated similar steps; iterated over image (two nested loops), changed some pixels value to 1 (made them white). I used the formula of ideal low pass filter (Figure 2). In the end, I reversed the values by making a subtraction to get ideal high pass fiter.

This time H was different, the middle circle was black whereas the rest of the image was white. It was vice-versa of the ideal low pass filter.

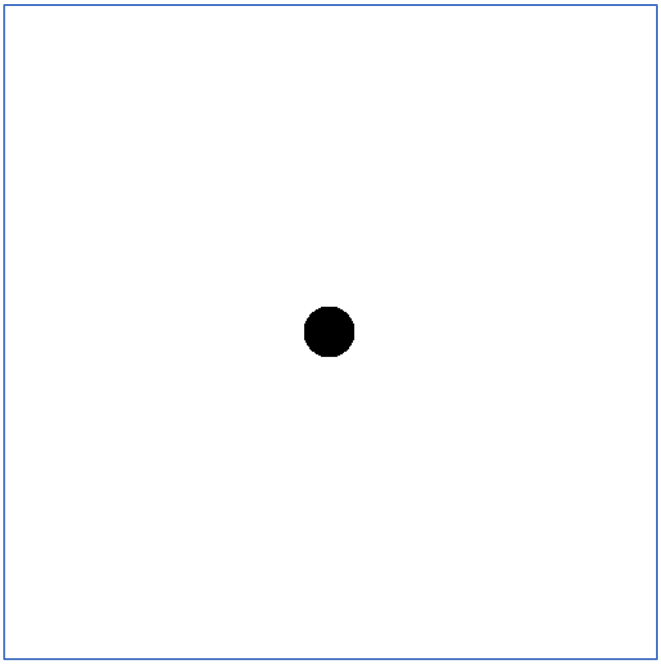


Figure 17: *H (Ideal high pass filter, cutoff = 20)*

Note: The image is bordered to make it noticable from white background (paper)

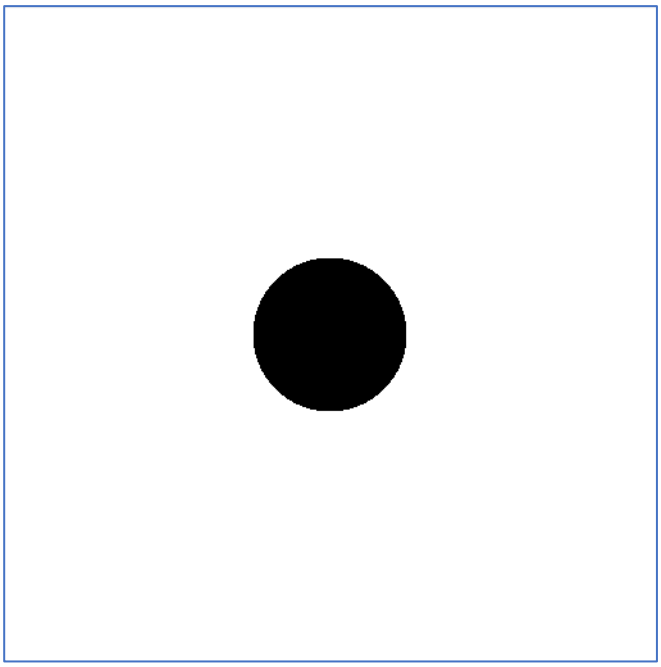


Figure 18: *H (Ideal high pass filter, cutoff = 60)*

By checking figure 17 and 18, we can realize that the size of the circle in the middle is related to cutoff frequency value. The size of the circle is directly proportional to the value of the cutoff value.

After that, I put zero freq in the upper left corner. Then I transformed my image with fft2 MATLAB's built-in function; I multiplied filter by Fourier image to apply the filter then did inverse FFT. I got following outputs with two different cutoff frequency values. I used abs function to show the image.

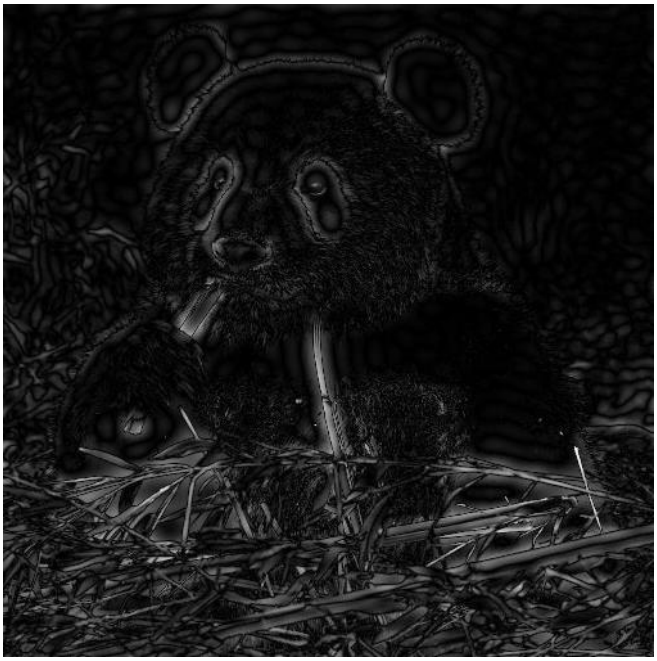


Figure 19: Ideal high pass *output image (cutoff = 20)*

Figure 20: Ideal high pass o*utput image (cutoff = 60)*

If we compare, figure 19 with figure 20 we see that, figure 19 is more sharpened. We can conclude that lower cutoff value results sharper image. Yet another, we have ringing in this filter. Ringing can bee seen more with less cutoff value filtered images.

b) Butterworth High Pass Filter

For this part, I followed the same steps as I did on the part 1-b. The only difference was reversing the filter by making the subtraction after applying the formula in Figure 6. I got the following filters below.

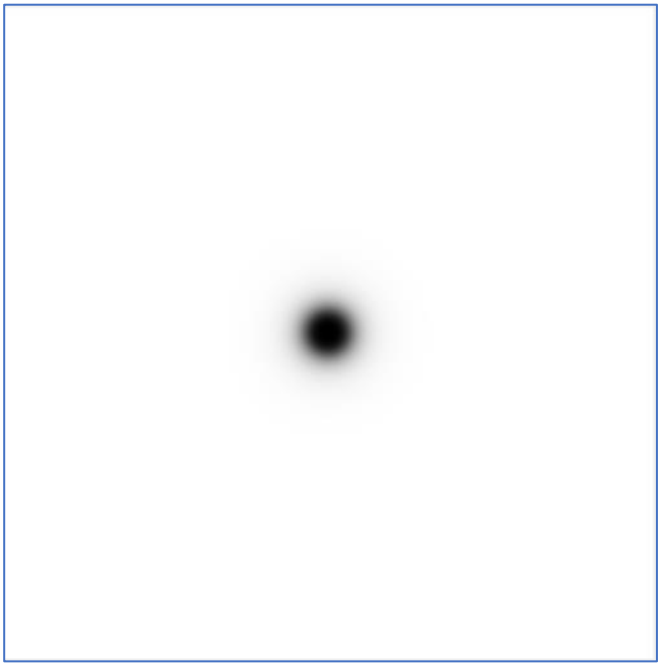


Figure 21: *H (Butterworth high pass filter, cutoff = 20)*

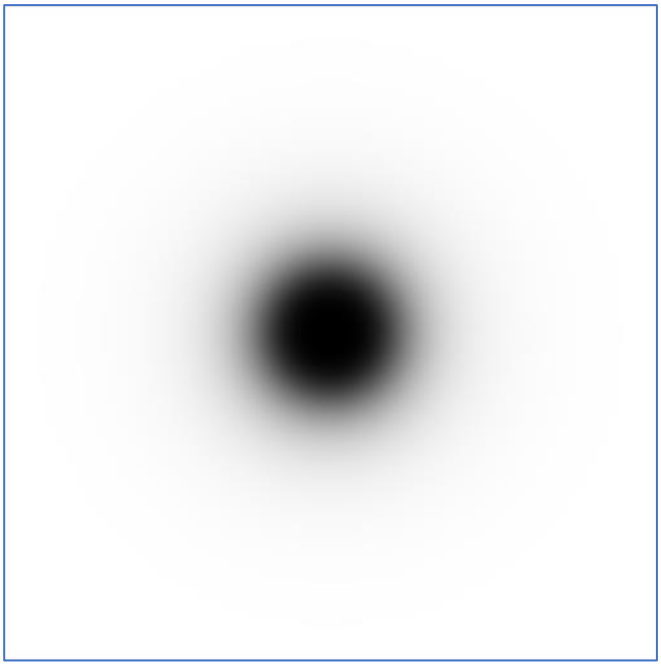


Figure 22: *H (Butterworth high pass filter, cutoff = 60)*

If we check figure 21 and figure 22, we notice that they are different compared to figure 17 and figure 18 because Butterworth filter keeps some exceptional frequency information.

After applying filter with different cutoff frequencies I got the those outputs:

Figure 23: *Butterworth high pass filtered output image (cutoff = 20, order = 2)*

Figure 24: *Butterworth high pass filtered output image (cutoff = 60, order = 2)*

By checking figure 23 and 24, similarly we see that the lower value of cutoff means sharper image. Thus, we can conclude that figure 23 is sharper than figure 24. Also contrary to ideal low pass filter, here we have no ringings because filter edges are soft.

c) Gaussian High Pass Filter

To complete the first part, I followed the same steps as I did on the previous parts. The only difference was again on the use of the formula since Gaussian formula is different. I arranged the filter image H by following thie formula in Figure 11. Then, in order to obtain Gaussian high pass filter I made a subtraction for the reversing purpose.

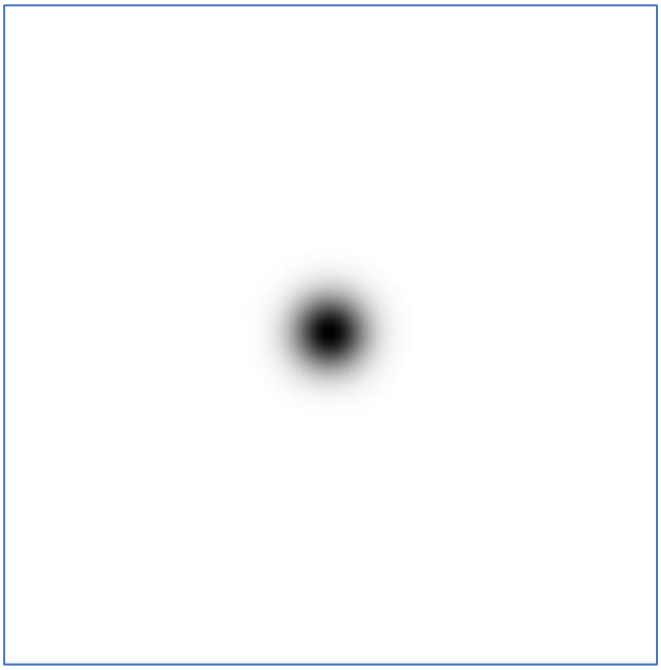


Figure 25: *H (Gaussian low pass filter, $\sigma = 20$)*

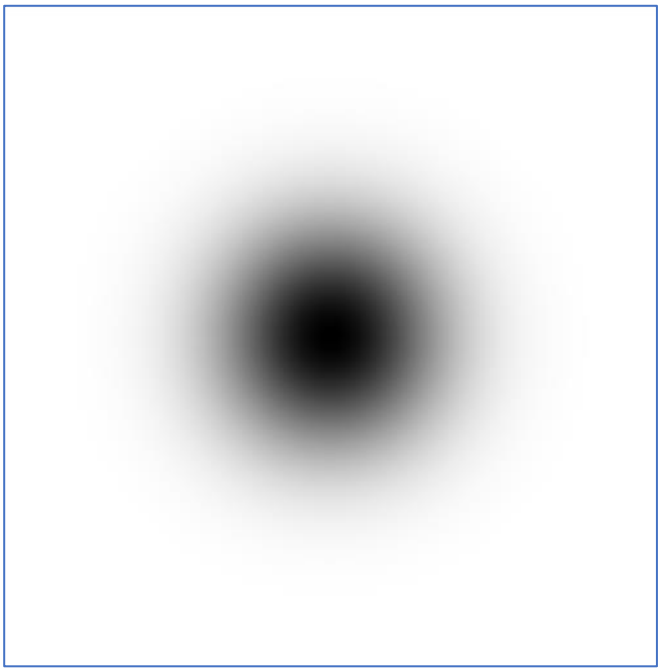


Figure 26: *H (Gaussian low pass filter, $\sigma = 60$)*

If we check figure 25 and figure 26, we notice that they are different compared to figure 17 and figure 18 because Gaussian filter keeps some exceptional frequency information similar to Butterworth.

Figure 27: *Gaussian high pass filtered output image ($\sigma = 20$)*

Figure 28: *Gaussian high pass filtered output image ($\sigma = 60$)*

If we make comparison between figure 27 and 28, again we realize that figure 27 is sharper than figure 28. Hence, we can deduct that the lower gamma value means sharper image we get. Also, contrary to ideal pass filtering, there is not any ringing here as the filter's edge are softer

PART 3: Selective Filtering - Notch Filters

For the last part of the homework, we have given 3 input images which are noised with Moire patterns. Our aim was removing the noise by using notch filtering strategy.

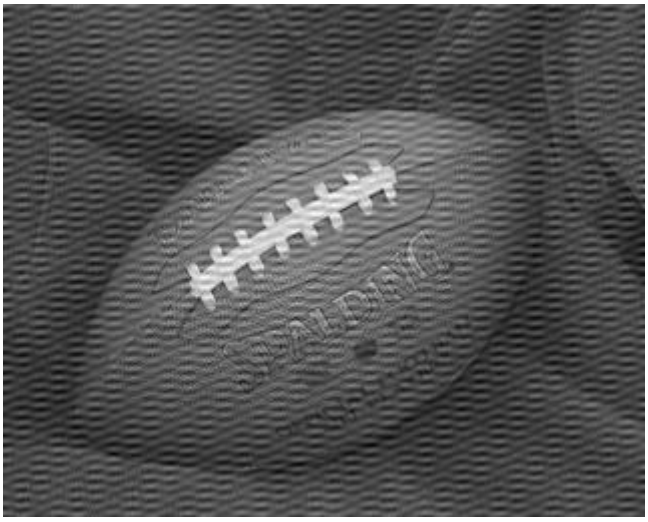


Figure 29: *Input image 1 (ball.png)*

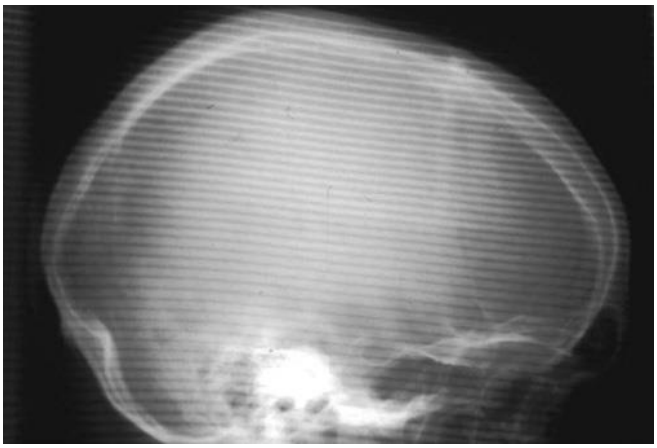


Figure 30: *Input image 2 (skull.png)*

Figure 31: *Input image 3 (bone.png)*

To complete this part of the assigment, first I read the input images. Then transformed them to frequency domain with MATLAB's built-in function fft2 with fftshift function. After that, to find spikes (noises) I calculated log of filter. Then I save this image with imwrite function, edited the image on application by placing black dots to spikes.

After successfully creating the filter, I applied notch filtering to supress Moire patter in my image. As I did in the previous parts, I multiplied my filter with image. In the end of my program, I applied inverse FFT with ifft2 function. As a result, I was able to construct clearer images with this technique. The Moire pattern was successfully removed.

The outputs of my program as follow respectively:

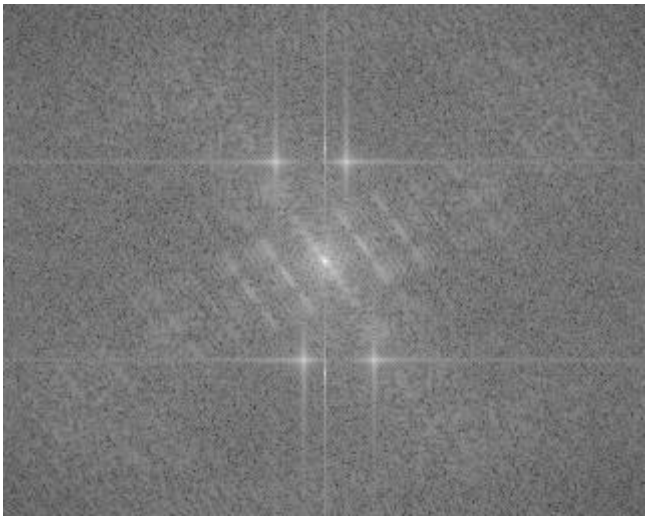


Figure 32: *Log mag of image 1 (ball.png)*

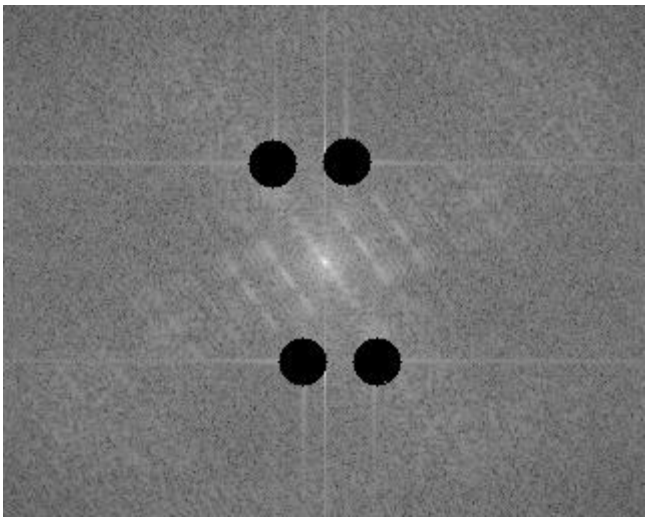


Figure 33: *Supressing Moire pattern (ball.png)*

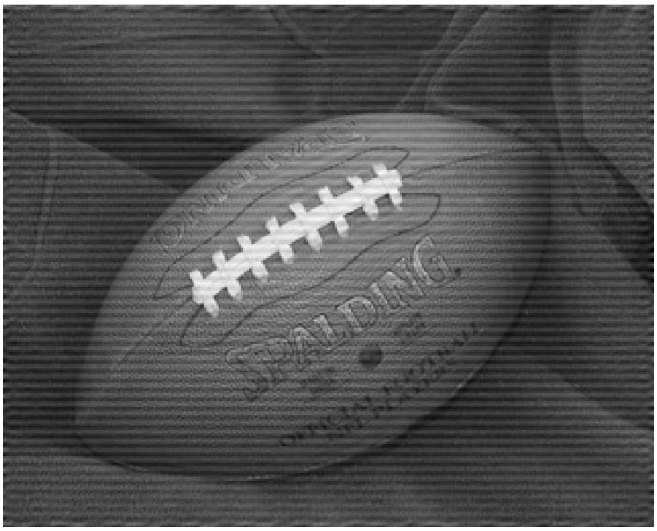


Figure 34: *Constructed image (ball.png)*

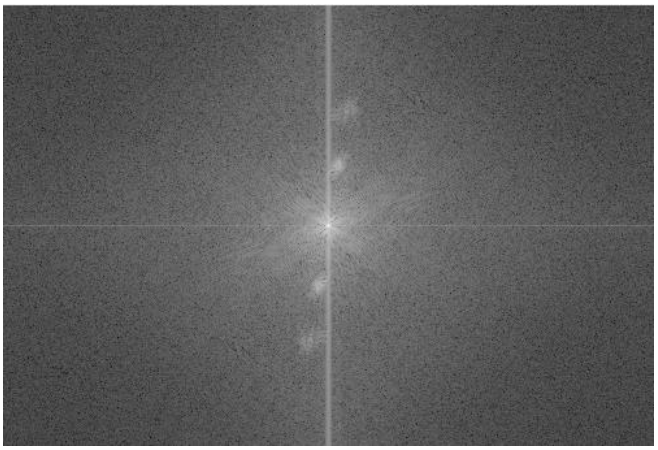


Figure 35: *Log mag of image 2 (skull.png)*

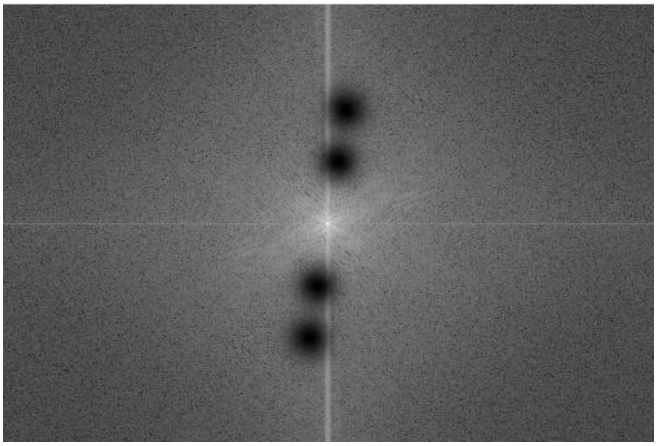


Figure 36: *Supressing Moire pattern (skull.png)*

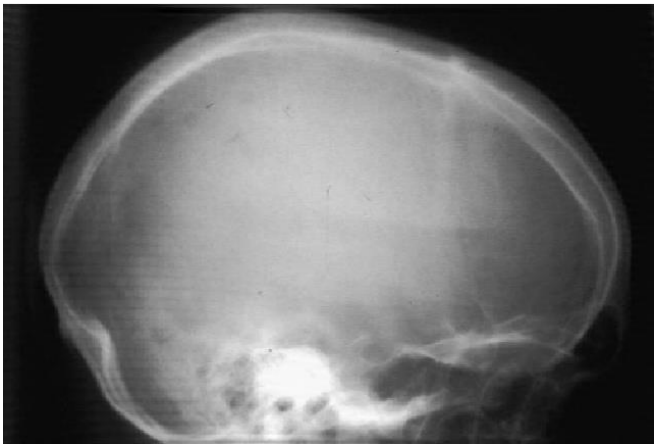


Figure 37: *Constructed image (skull.png)*

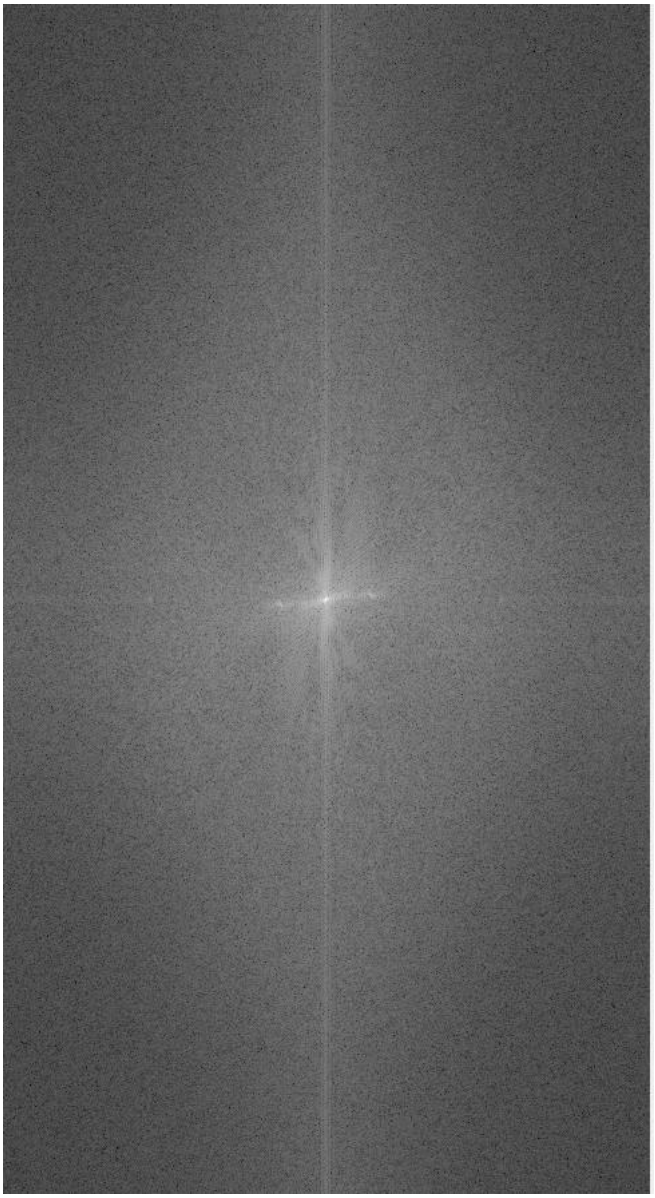


Figure 38: *Log mag of image 2 (bone.png)*

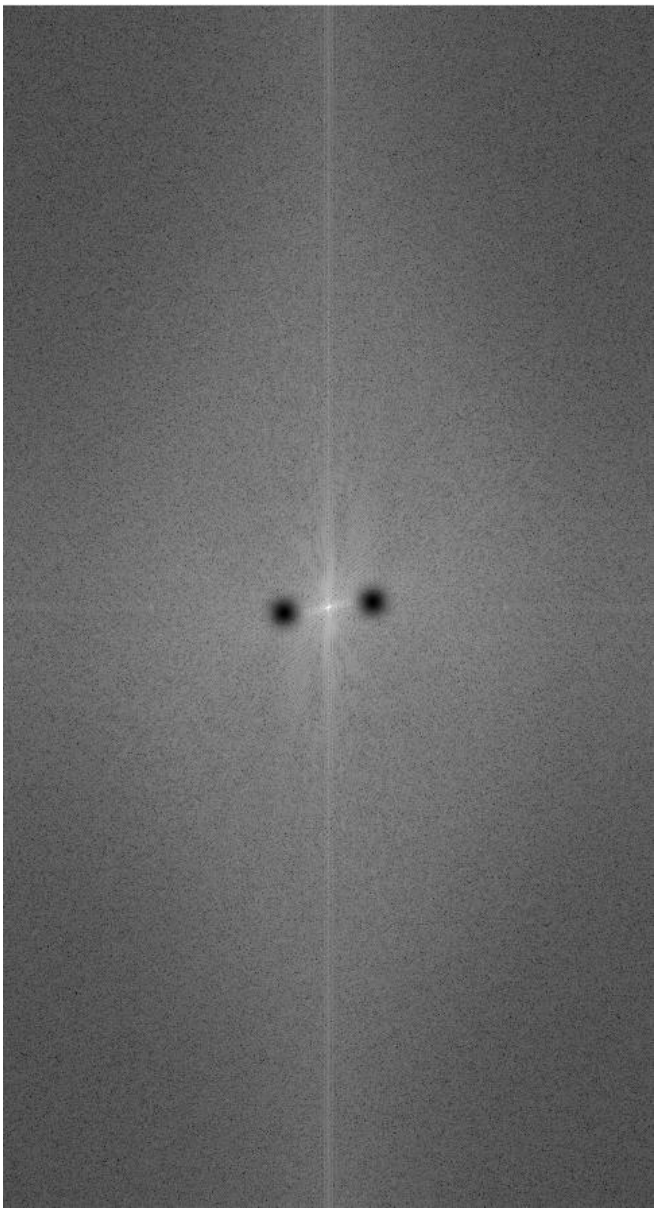


Figure 39: *Supressing Moire pattern (bone.png)*

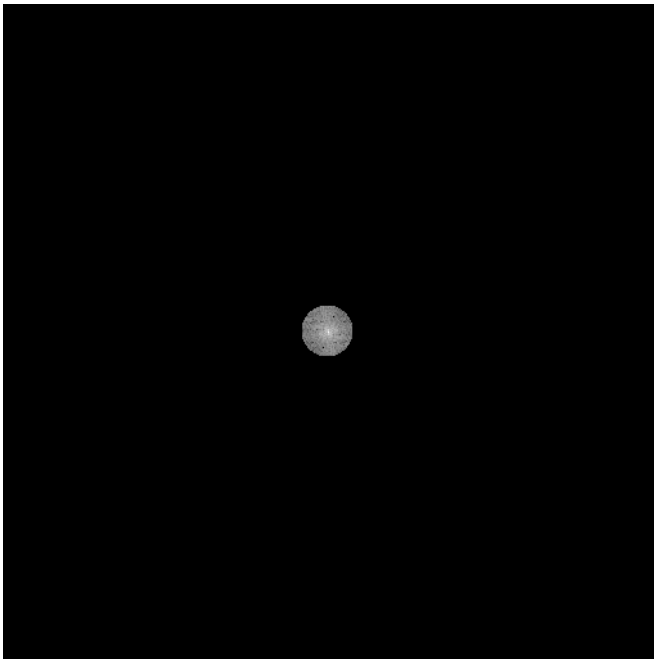
Figure 40: *Constructed image (bone.png)*

If we compare figure 34, 37 and 40, we see that figure 37 and 40 are more clear on removing the noise which means figure 34 still a little bit noisy even after notch filtering. It is related to input image itself. However, notch filtering is quite good algorithm for the removal of the Moire patterns in the images.

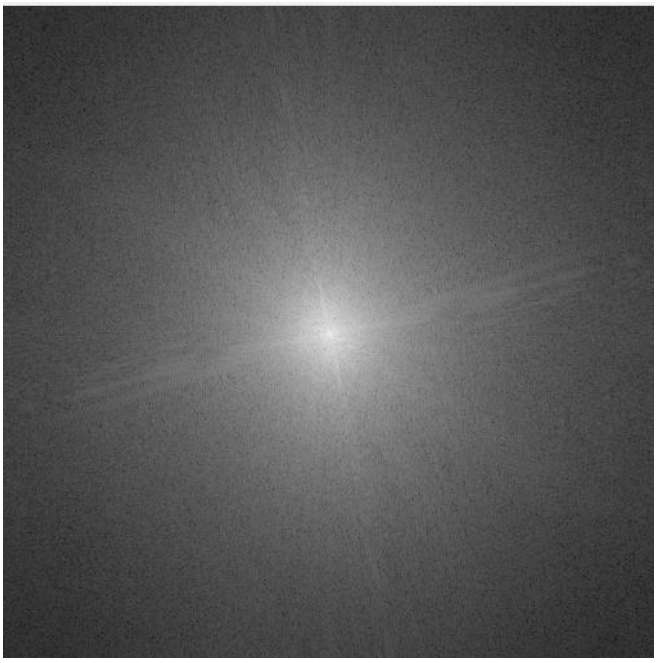
III- APPENDICIES

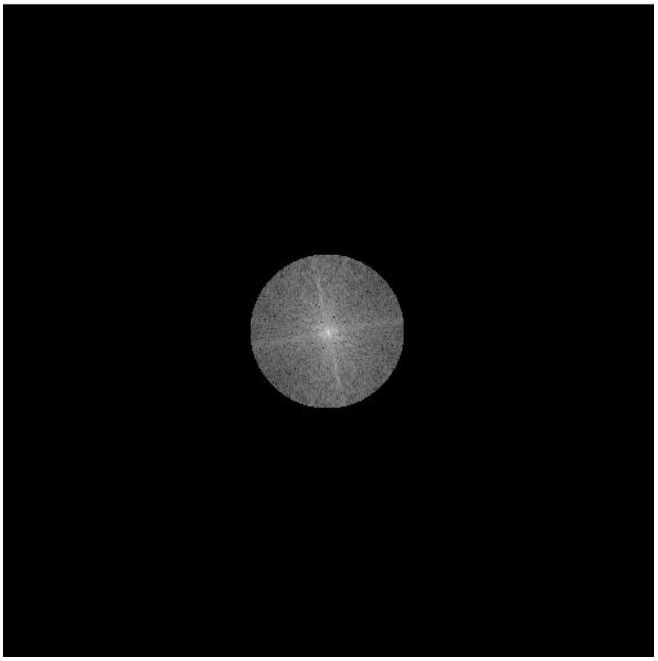
Results in Frequency Domain

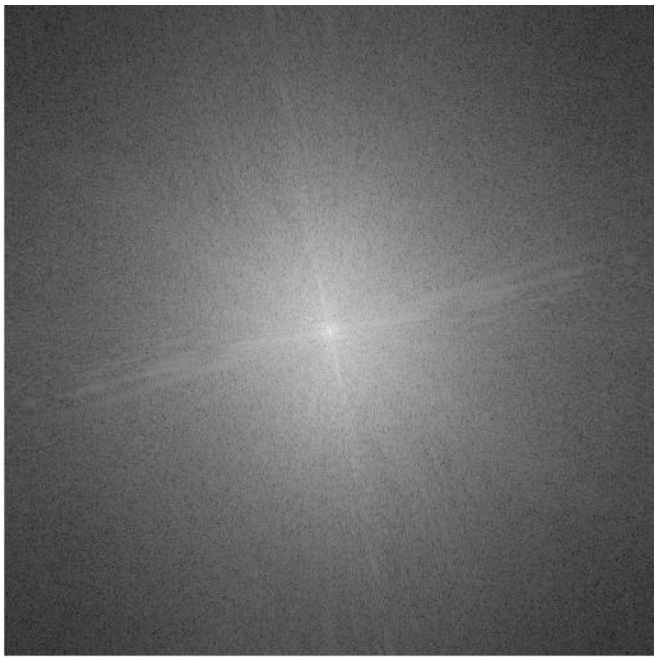
Part 1-a (cutoffs: 20,60)



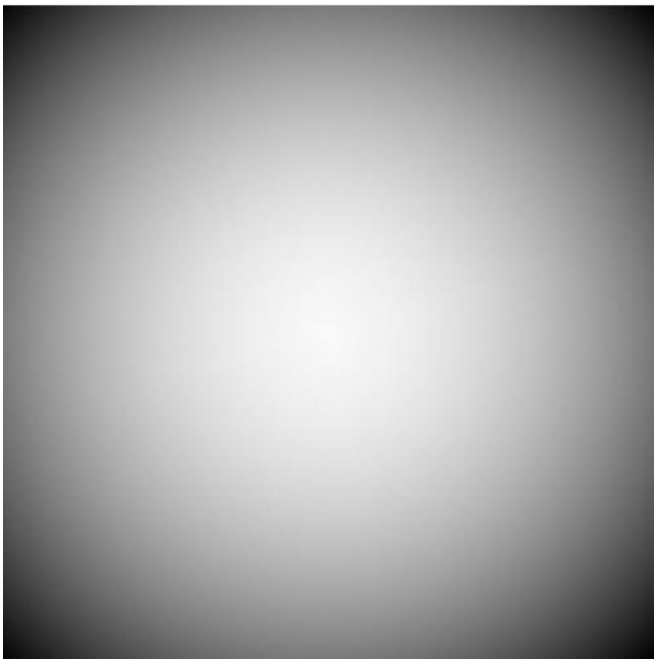
Part 1-b (cutoffs: 20,60)

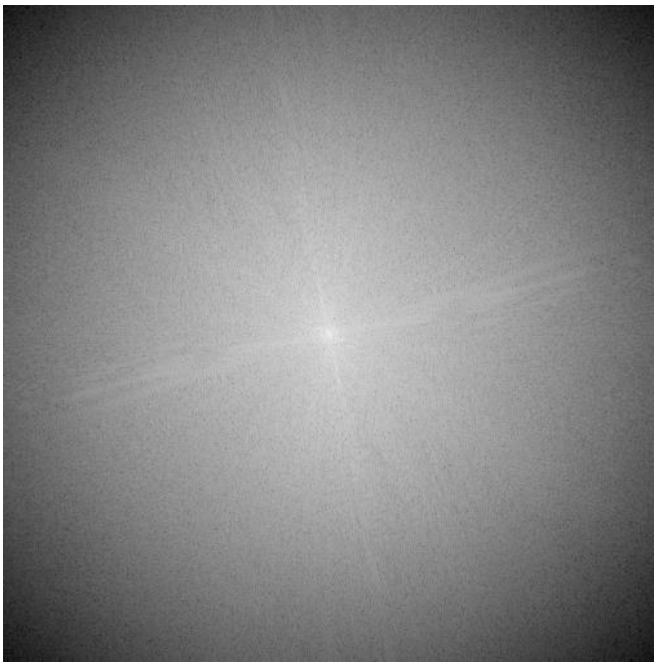




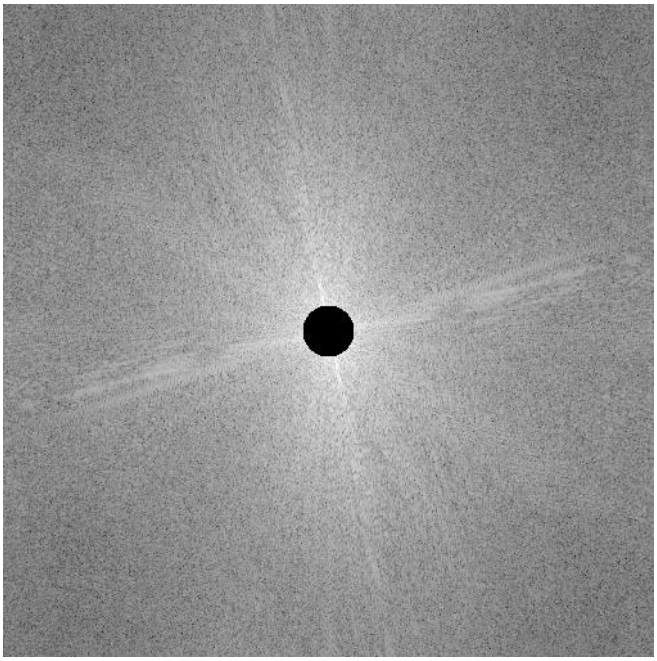


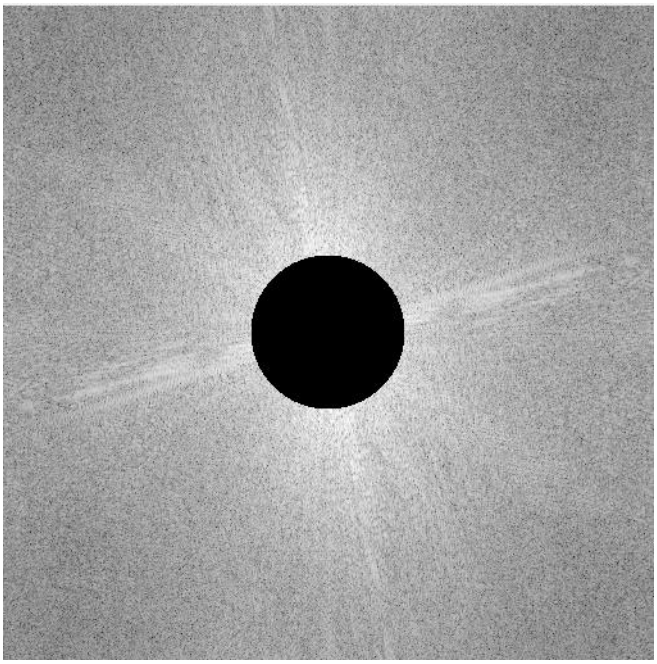
Part 1-c (cutoffs: 20,60)



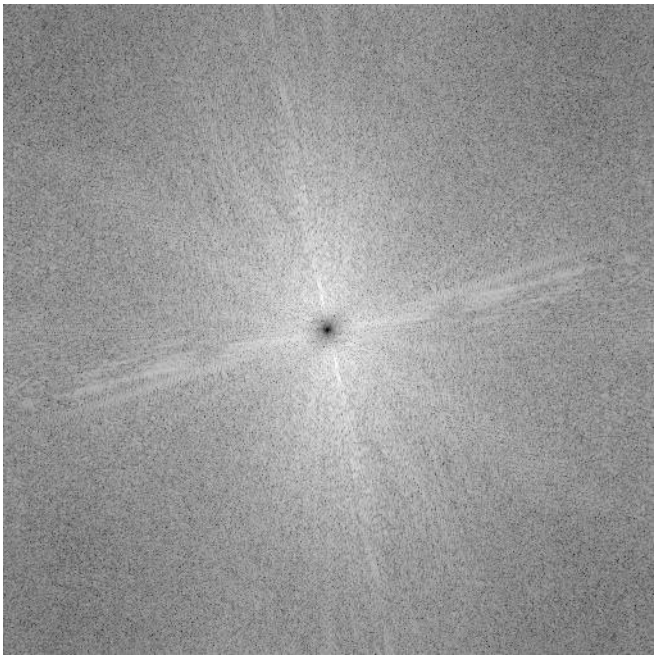


Part 2-a (cutoffs: 20,60)





Part 2-b (cutoffs: 20,60)



Part 2-c (cutoffs: 20,60)