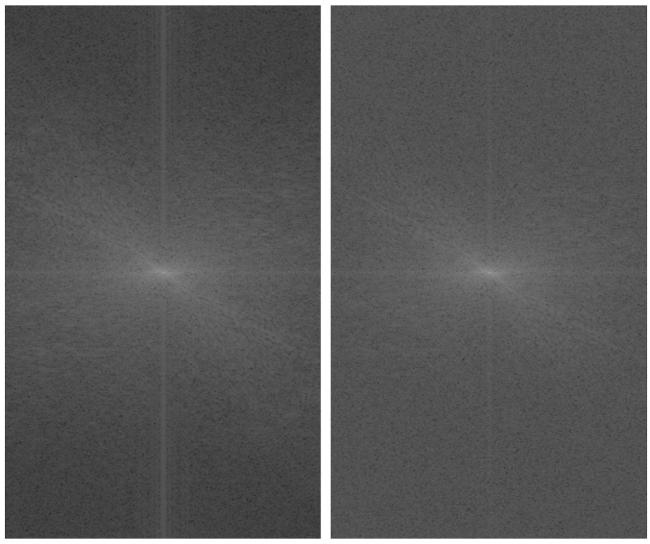
COSC - 6380 Digital Image Processing
ASHNA SHAH
Assignment -3

1) Implementation of Discrete Fourier Transformation
    a) While implementing the formula
        - After initializing the fft_matrix, we loop the matrix first N*N times and then initialize the result variable to put the value at that place in our dft matrix
        - After that we loop again through the matrix and fill the obtained value by the given formula (results in complex number) , at the respective position.
        - Time complexity is N^4.
        - In the **inverse dft** we do everything the same way as above except that the input matrix is dft matrix unlike computing dft has input matrix as image.
        - To find the **cosine dft** we compute only first part of the formula from dft computation which is the real part. The other part is imaginary or sin part.
        - To compute **magnitude** we need to take real and imaginary parts separately so that we can compute the formula.

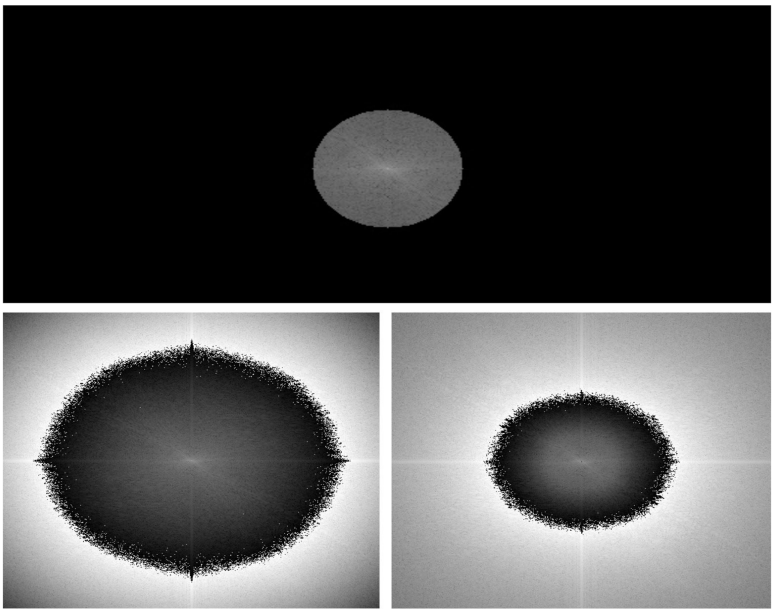
2) Filtering Implementation
    a) By computing fft on the image using inbuilt function np.fft.fft2(), we make a shift in the image to get lower frequencies in center and generate magnitude of that. Finally, convert it to grey_scale and showing it as output[0].
        i) The DFT image for all filters either low or high remains same as shown below. It changes when the same image is either coloring or B&W.
        ii) In the image on left side the DFT shown is for color image which is more darker than on right it is for B&W image which is more brighter.

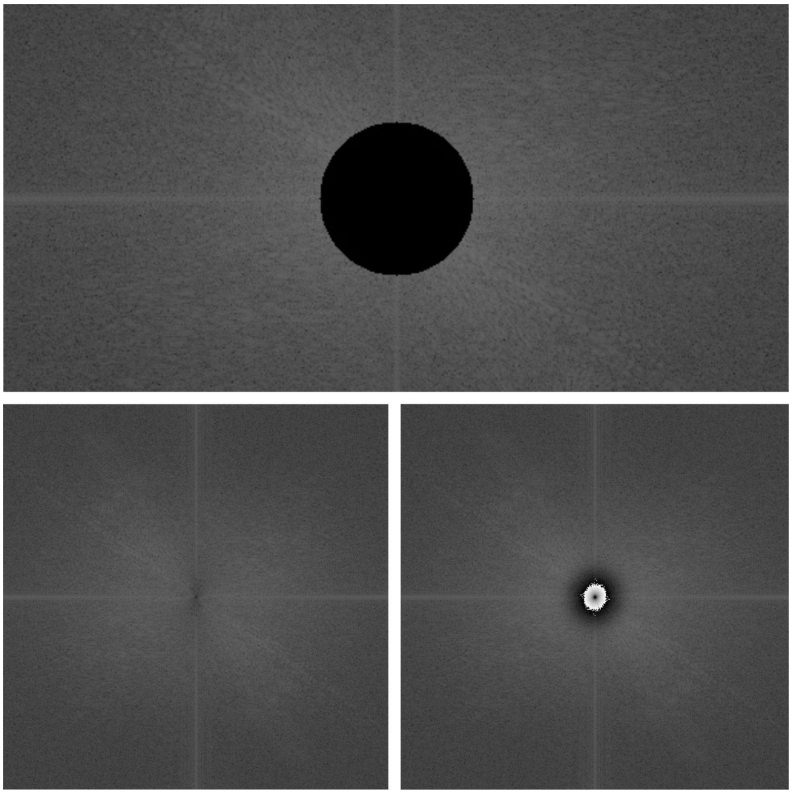


b) After this, we compute filtered image using convolution theorem, by multiplying the shifted fft image and the mask generated from the respective method. We generate magnitude of the filtered image. Hence, converting it to grayscale we get another image to save.

In this filtering it doesn't matter where it coloring or B&W, what matters is whether it is high or low pass. First collage is for low pass and second collage is for high pass.

- The horizontal in both images is for ideal_filter. In Ideal_low and Ideal_high if the cutoff is same, it exactly does the opposite image for both of them.
- In second row on left, is for gaussian_filter. In gaussian_low_pass the filtered image would be more brighter on the edges than in gaussian_high_pass, where we can see that there is one black dot at the center and rest of the image becomes smoother as cutoff increases.
- In second row on right, it is for butterworth_filter. In butterworth_low_pass, order is 5 so we can see the black circle is bigger than it would have been if order is 2. Also, in butterworth_high_pass, order is 2 and it is exactly opposite of low_pass_butterworth, so we get white circle which gets bigger as the order increases.



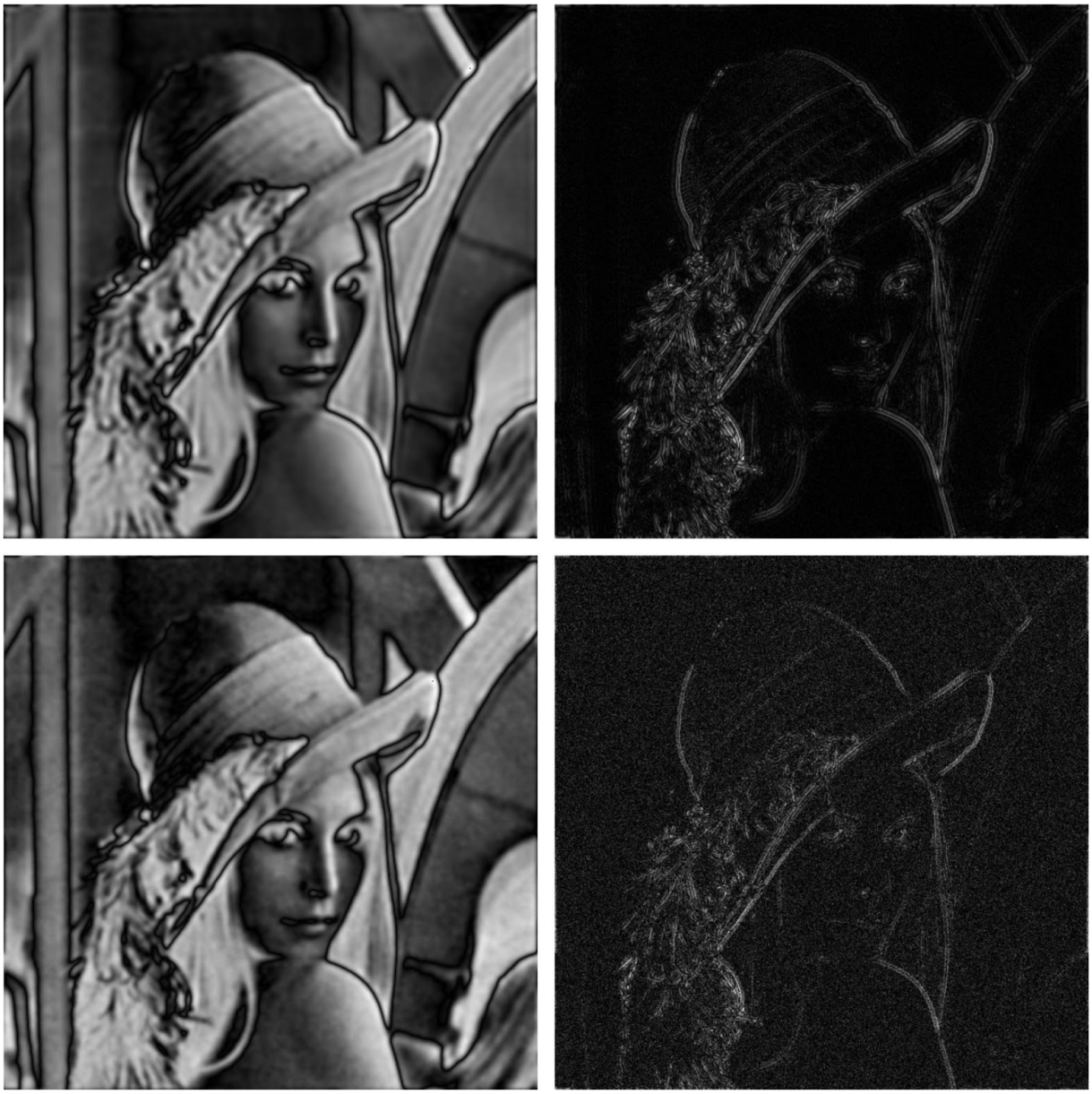


c) Finally, we do inverse shifting fft of the filtered image, on that we perform inverse fft and compute magnitude of it. Passing the last image to full contrast stretch we get our final image. In full contrast I did not compute negative as I computed the fft with numpy and cv2.

In the collage below, there two images in first row labeling from left as 1(low pass) and 2(high pass), which shows result for coloring image. In second row we start labeling from left as 3(low pass) and 4(high pass) shows result for B&W image.

1. Ideal Filter
    a. In the results for coloring image, in low pass we could see many ring like structure on her face and shoulder but is smoother. While in high pass also, it occurs but in black color and is sharper than low pass.
    b. For b&w, we can see above characteristics, but also there is more salt and pepper in both images than the coloring image results.

2. Butterworth Filter
    a. For coloring image results order is 5. So as order increases, for low pass blackness increases while for high pass whiteness increases.
    b. For b&w image results, order is 2. So we can see the difference described above. Also right now cutoff is 50,but as the cutoff increases, the image becomes more clearer than compared to present.



3. Gaussian Filter
    a. As we can see, so far we have got the best result in gaussian filter. At present cutoff is 50, but as it increase we can get the same image as uploaded. It is smoothest of all three filters.
    b. For b&w image results, it is same as ideal there is more salt and pepper noise for both low pass and high pass than the colored image results.

*Note: I have compressed the magnitude using log(abs) and then multiplied it with 20. I first tried with 10 but results are better visible with 20.