Ceng 466 - Take Home Exam 2

Eda Özyılmaz
2171882

Hilal Ünal
2172112

I. Introduction

This document is designed as a report for CENG466, Fundamentals of Image Processing, Take Home Exam 2. This report includes explanations of our methodology and analysis of the results we received while developing our code. The report mainly explains the methodology and rationale behind our algorithm design. Comments and explanations can be found in the following section.

II. Questions

In this homework, there are three questions which are Question 1 - Noise Reduction using Frequency Domain Filtering, Question 2 - Edge Detection, and Question 3 - Image Compression with Wavelet Decomposition. The questions and our answers, analysis, and methods to solve these questions are explained in detail in the following sub-sections.

A. Question 1 - Noise Reduction using Frequency Domain Filtering

In first question, we were given noisy images and we identified the noise types of these images and tried to remove the noises using Frequency domain filtering to recover the original image. We used the following Matlab code to get the given images :

$$a1 = imread('./A1.png');$$

First, to detect noises on given images we used **fft2(..)** function as stated in the homework pdf. This function takes the Fourier Transform of the given images. Taking Fourier Transform of an images converts signal in spatial domain to a signal in frequency domain. Basically, it changes the images basis to frequency domain. To shift the Fourier Transform, we used **fftshift(..)** function. This function shifts zero-frequency component to center of spectrum. After taking the Fourier Transform of the given images and shifting it, we write the images by using the **imwrite(..)** function. The Matlab codes for getting Fourier transform and shifting are below:

$$a1\_fourier = fft2(a1);$$
$$a1\_shifted = fftshift(a1\_fourier);$$
$$ab = abs(a1\_shifted);$$
$$ab = (ab - min(min(ab)))./(max(max(ab))). * 255;$$
$$imwrite(ab,'Fourier\_a1.png');$$

The Fourier Transform of the image represents a plot of high and low frequencies. Te different frequencies represents noises. Therefore, we tried to eliminate the different frequencies on the Fourier Transform of the given images. When we eliminated the different frequencies on Fourier Transform we obtained the noiseless images. Each input image, their noise types and the output images are explained below.

For the first image A1.png, we obtained the following Fourier Transform which can be seen in the Figure 1. to obtained the Fourier transform version of the image, we used the code that explained above. This Fourier Transform has a white vertical line in the middle, this cause noise in the original image. Therefore, by applying filters we tried to get rid of this white line without changing the white area at the middle. Between the distance of the white area's ending from center of this Fourier Transform image, we applied Gaussian low Pass Filter which has the following formula:

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

D is the distance 20 which is the white area's ending from center, and our D_0 is 30.

For the other pixels, we applied Ideal Low Pass Filter. When the distance of the chosen pixel is larger than 20 and the y coordinate of the pixel equals to y coordinate of middle + 1, we made that pixel black, 0.

After traversing all of the pixels with the algorithm that explained above, we got the following result on the Fourier Transform image, which can be seen on the Figure 2. When we take inverse Fourier Transform, we obtained the noiseless version of the original image. To get the inverse Fourier Transform we used the following code lines:

$$a1\_output = ifft2(ifftshift(a1\_filtered));$$

This gave us the noiseless version of A1.png, A1_denoised.png.

For the image A2.png, we had the following Fourier Transform image which can be seen in the Figure 3. In this Fourier Transform, we had unwanted circles. We calculated the distance of these circles and tried to get rid of them by applying filter son the Fourier Transform to obtain noiseless version of the original image. The circles are between 280 and 320 , and between 62 and 88. We applied Ideal Low Pass Filter to turn these pixel into black, 0. The Ideal Low Pass Formula that we used can be seen on below:

$$If 62 < D < 88 || 280 < D < 320 then H(u,v) = 0$$
$$Else H(u,v) = 1$$

where D is the pixels distance from the center of this image. When we applied this filter, we got the Fourier Transform on

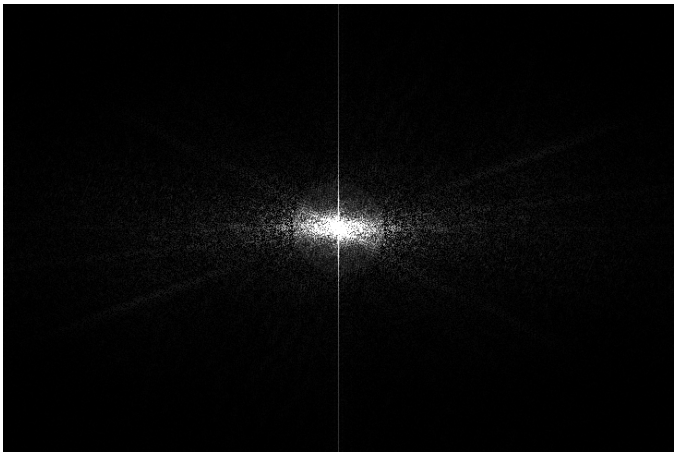

Fig. 1. Fourier Transform of A1.png

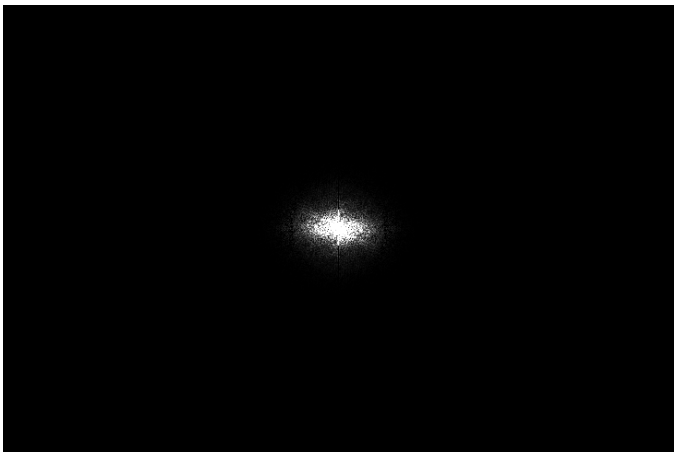

Fig. 2. Fourier Transform of A1.png with the filters

the Figure 4. When we take inverse Fourier Transform, we obtained the noiseless version of the original image.

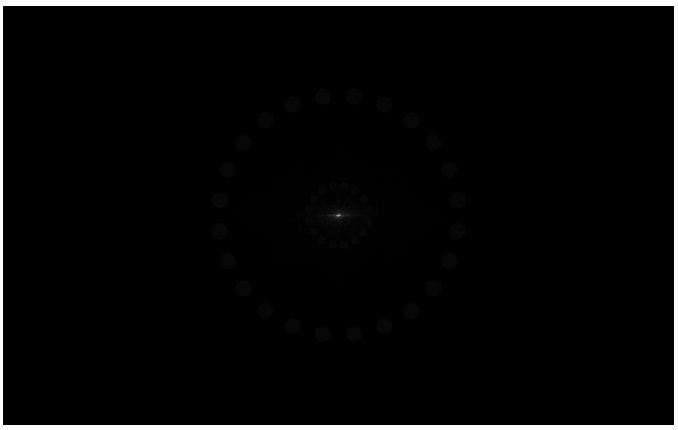

Fig. 3. Fourier Transform of A2.png

For the image A3.png, the obtained Fourier Transform can be seen in the Figure 5. We tried to get rid of the green and pink spots on this Fourier Transform to obtain noiseless image.

Fig. 4. Fourier Transform of A2.png with the filters

therefore we applied Ideal Low Pass filter. We first found the distances of these spots, and then applied filter to turn them into black, 0 to get rid of the noise in the original version of the image. When our Ideal Low Pass Filter is applied, we obtained the Fourier Transform which can be seen in the Figure 6. after applying inverse Fourier Transform, we got the noiseless image.

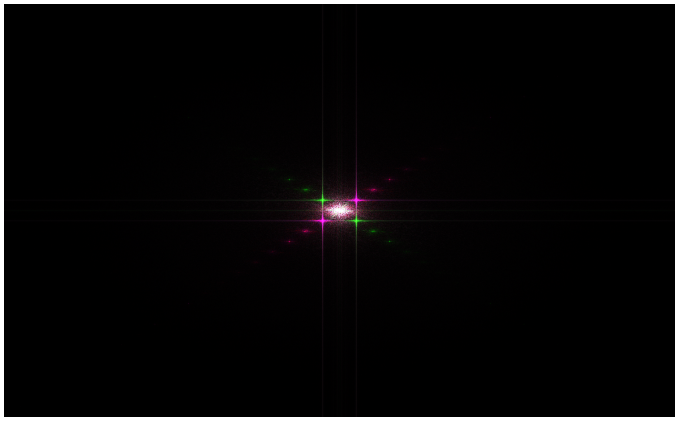

Fig. 5. Fourier Transform of A3.png

B. Question 2 - Edge Detection

At this question on the homework, we were asked to find edge maps of the given two images by using edge detection in Fourier Domain. Edge detection filters are used to detects edge maps of given images and find edge maps.

To find edges by using edge detection in Fourier Domain, we first obtained the Fourier Transform of the given images by using the following codes:

$$b1 = imread('./B1.jpg');$$
$$b1\_shiftf = fftshift(fft2(b1));$$

As explained in the *Question 1*, **fft2(..)** takes the Fourier Transform of the given image and **fftshift(..)** function shifts zero-frequency component to center of spectrum.

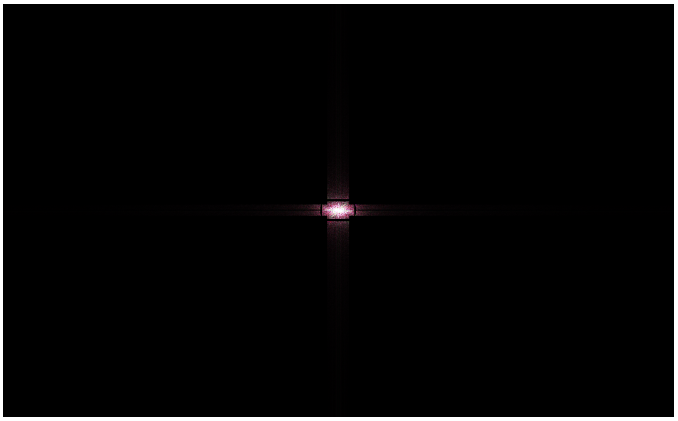

Fig. 6. Fourier Transform of A3.png with the filters

For image B1.png, after getting the Fourier Transform of the image we applied the following Butterworth High Pass Filter:

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

where D_0 is 30, D is the distance between the pixel and the center of the image, and the n is the order which equals to 1. After applying this filter and taking the inverse Fourier Transform, we obtained the edge map pf the B1.png.

For image B2.png, after getting the Fourier Transform of the image we applied the Butterworth High Pass Filter above with different values than B1.png, where D_0 is 40, D is the distance between the pixel and the center of the image, and the n is the order which equals to 2. After applying this filter and taking the inverse Fourier Transform, we obtained the edge map of the B2.png.

C. Question 3 - Image Compression with Wavelet Decomposition

In this part of the homework, we were expected tho compress given images with using wavelet decomposition and then compare our results with JPEG format using mean square error formula.

We use built-in MatLab functions **dwt2(..)** and **idwt2(..)** to calculate wavelet transformation.
dwt2(..), takes two parameters, image as an input data and wavelet as a decomposition type. It returns four parameters as $[cA, cH, cV, cD]$, cA is a approximation coefficients matrix and $[cH, cV, cD]$ are detail coefficients matrices that represents horizontal, vertical, and diagonal, respectively.
idwt2(..), takes five parameters $[cA, cH, cV, cD,' wname']$. $[cA, cH, cV, cD]$ are same as the output of **dwt2(..)** and *'wname'* is decomposition type. It returns an image as an output for our code.
We decided to use Haar wavelet as a decomposition type in our project.

In this part of the homework we write two more additional files to write functions, one is for compression and other is for decompression. Both of the functions take two parameters first is for input image and the second is for output image.

In both compression and decompression functions, we first take the wavelet decomposition of the input image, after that we only use output approximation coefficients matrix, *cA*, to compress the image, since *cH, cV* and *cD* are detail coefficient matrices.
In both functions, to compress the image and decompress the compressed image, we multiply *cA* with quantization number, convert the output to *uint8*, divide it to the quantization number, and convert the output to a double.
After trying numbers between 1 and 255, we decided that we get the best output, smaller compressed image size and closer decompressed image size to the original image size, when we set the quantization number as 50.
After compression and decompression we use **idwt2(..)** to composed the images.
In both functions we return the size of the output images to be able to compare the size of this images to size of the input images.

Even though we planed to use Huffman encoding and Huffman decoding between compression and decompression, we cannot make the code work when we try to apply Huffman encoding and decoding.
Aim of our Huffman coding was to be able to implement lossless compression rather than lossly compression by creating binary tree of nodes and removing the highest priority nodes hence decrease the size of the image.
But unfortunately we cannot apply this method to our code properly.

After getting the size of the output images from the compression and decompression functions, first we compare size of the input image and the compressed image using compression ratio. Results we got for compression ratio is between is between 4.0 and 8.0, we read this as we manage to compressed the image nearly between 4-8 times.
Secondly we compare the size of input image and decompressed image by using mean square error formula. We get results between 10^{-4} and 10^{-5} which are small enough for us to think that after decompression, size of output image becomes nearly same as the size of input image.

III. Conclusion

The aim of this homework is to understand how to denoise an image and detect its edge maps in Fourier domain, and also to learn compressing and decompressing an image with Wavelet decomposition.

In question 1, we get the Fourier Transforms of the given images and applied appropriate low pas filters to denoise them. To denoise the Fourier Transform, we tried to get rid of the different spots on the transform. Then we took the inverse Fourier Transform the obtain noiseless version of the original image.

In question 2, we again get the Fourier transforms of the given images. And to obtain edge maps of these images we applied appropriate high pass filter to these Fourier Transforms.

After applying filter, we took the inverse Fourier Transform and obtained the edge maps of the given images.

In question 3, we compress the images with wavelet decomposition. We only use quantzation and dequantization to compress and decompress the images. The quality of the output image is not as good as and the size difference between input image and output images as successful as it is expected to be when using Huffman coding, but we get to compress the images 4 to 8 times and able to decompress them nearly as much.

REFERENCES

[1] Seniha KETENCI, Ali GANGAL. (2017, May). Automatic reduction of periodic noise in images using adaptive Gaussian star filter. [Online] Retrieved December 2019, from https://dergipark.org.tr/tr/download/article-file/433724

[2] Nayak, J. (n.d.). Retrieved December 4, 2019, from https://universe.bits-pilani.ac.in/uploads/JNKDUBAI/ImageProcessing7-FrequencyFiltering.pdf.

[3] Single-level discrete 2-D wavelet transform, from https://www.mathworks.com/help/wavelet/ref/dwt2.html