CENG466 - Take Home Exam II

Berk Güler
2310092

Onur Demir
2309870

I. INTRODUCTION

This document is prepared for CENG466 - Take Home Exam II. This report includes the solutions of the given problems, and also explains our methodology and includes the analysis off the results and our comments on them.

II. QUESTIONS

In this homework there are 2 parts and 3 questions given. First part is about Frequency Domain Image Filtering whereas second part is about JPEG Image compression. In the first question, implementing an edge detection algorithm in frequency domain is asked. In the second question it is asked that enhance given images in Fourier domain by using denoising techniques. This question was not generic for all photos, it is specific for given 2 images. Finally, in the last question it is asked that to implement JPEG image compression by converting color space, applying DCT and Huffman Coding.

A. Question 1 - Edge Detection

Firstly, we read the image in gray scale format in order to detect edges. In frequency domain we use high pass filter to detect edges. In this question we decided to use Butterworth High Pass Filter. We can find Butterworth High Pass Filter with the given formula:

$$H(u,v) = \frac{1}{1+[d_0/D(u,v)]^{2n}}$$

where $D(u,v) = [(u - P/2)^2 + (v - Q/2)^2]^{\frac{1}{2}}$

We chose the r as 80 and n as 3 then, find the butterworth filter that we are going to use.

In order to use Butterworth High Pass Filter we transform the image to frequeny domain by using np.fft.fft2. Then, by using np.fft.fftshift we centralized the frequencies. After that, since we can use multiply in frequency domain like convolution in spatial domain, we multiply the butterworth filter with out transformed and shifted image. After applying filter, we first shift back the frequencies to the corners by using np.fft.ifftshift. Then by using np.fft.ifft2, we tranformed our image from frequency domain to spatil domain. We took the abs of that image since there are some negative values in the image array after transform operations.

If we compare the results with the spatial domain filters we can say that edge detection in frequency domain is much more easy since we can just multiply the image with filter after transform operations, and transform operations are exists in the numpy library. Also, we think that results that we got in this homework are better than the the1 results, since there are less noise in the image and edges are more evident.

B. Question 2 - Noise reduction

In this part we tried some filters to denoise the given noisy images. We chose the filters from our lecture notes and from the internet. After trying different filters we decided to use Gaussian Blur for enhance_3 and Butterworth filter for enhance_4.

- Enhance - 3
  Firstly, we read the image in gray scale format. Then we split the image to their channels and applied Discrete Fourier Transform for 2D by using "np.fft.fft2". However we needed to rearrange the channels for a better visual output to analyse the frequency in Fourier Domain. So we shifted the channels by using "np.fft.fftshift". Since convolution in time domain is equal to the multiplication in frequency domain, we multiplied the shifted rgb channels with Butterworth LPF (with r 50 and n 0.5). We analyzed the output image in frequency domain (see Fig. 1.). We tried to reduce the Gaussian noise in

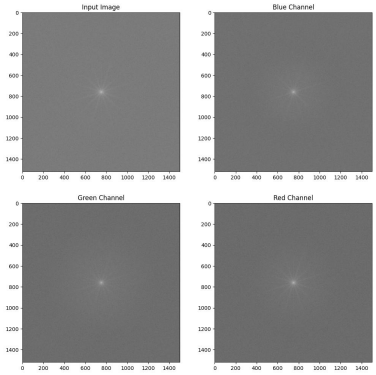

Fig. 1. Image in Fourier Domain

the frequency domain using Butterworth LPF but the result is not very effective. After this step, we decided to reduce the Gaussian noise in the frequency domain using Gaussian LPF (with r 50). Then we applied inverse shift to the channels and took absolute values. Finally we merged the channels by using "cv.merge" to get the output image. Finally, we see that noise reduction in

frequency domain is better since we can select the desired frequency area by looking through the Fourier Domain output image.

- Enhance - 4
  Firstly, we read the image in gray scale format. Then we split the image to their channels and applied Discrete Fourier Transform for 2D by using "np.fft.fft2". However we needed to rearrange the channels for a better visual output to analyse the frequency in Fourier Domain. So we shifted the channels by using "np.fft.fftshift". Since convolution in time domain is equal to the multiplication in frequency domain, we multiplied the shifted rgb channels with Butterworth LPF (with r 50 and n 0.5). We analyzed the output image in frequency domain and see that we can use such a mask (see Fig. 2.). We tried to

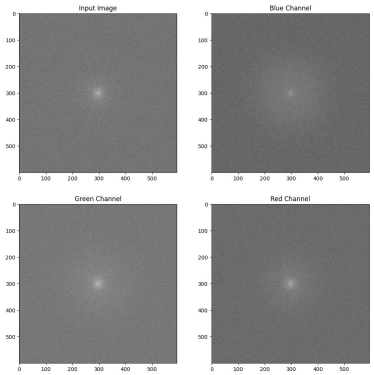

Fig. 2. Image in Fourier Domain

reduce the Salt and Pepper noise in the frequency domain using Butterworth LPF. We also used Ideal LPF for these part however Butterworth LPF was better than Ideal LPF since it shows the edges more precise than Ideal LPF After this step, we decided to reduce the Salt and Pepper noise in the frequency domain using Butterworth LPF (with r 50 and n 0.5). Then we applied inverse shift to the channels and took absolute values. Finally we merged the channels by using "cv.merge" to get the output image. In the first homework we were used Median filter for salt and pepper noise however we could not use it in these homework since it is non-linear filter. Finally, we see that noise reduction in frequency domain is better since we can select the desired frequency area by looking through the Fourier Domain output image. However we can say that Median Filter also gives good results in time domain.

C. Question 3 - JPEG Compression

In this part, we are asked to compress a given image then show to user compression size and compressed image. This part contains two functions one for compressing and printing the compression size namely the2_write and the other one for decompressing the compressed image in the_2write and show it to user. We first read the image and get the shape of the image. Luminance is more important to the eventual perceptual quality of the image than color. So we convert from RGB color space to one where luminance is confined to a single channel. This color space is called YCbCr. Here, Y is the luminance component and Cb, Cr are the chrominance components. They are the blue and red differences respectively.

Since chrominance is not very important for human eyes, we can downsample and reduce the amount of color (Cb - Cr components). Cb and Cr components are reduced by factor of 2 in both directions (vertical and horizontal). Finally, Y is sampled at each pixel, whereas Cb and Cr are sampled at every block of 2x2 pixels. Now for every 4 Y pixels, there will exist only 1 CbCr pixel.

After downsampling we get subimages of blocksize 8x8 in order to use Discrete Cosine Transform. Since we used DCT, and coise waves go from -1 to 1, we centeralizedd our values around zero. We shift the range from 0:255 to -128:128 by substracting 128 from each pixel values. After that we take the Discrete Cosine Transform of these 8x8 blockes by using cv.dct library function. We made this operations for every channel of image namely, Y, Cr and Cb. Then we quantized the coefficient table we obtained using DCT. This is the real lossy part of the process. In the table of coefficients we got through DCT, the top-left cells refer to low frequency part, and the bottom-right cells refers to high frequency part. We know that high frequency part can be eliminated without much loss in the look of the image. Every value in the coefficient table is divided by the corresponding value in the quantization table and rounded to the nearest integer.

After that we decided to decompress the image without using Huffman encoding. We applied inverse of the compress operation to get the image back. Multiply with the quantization table, add 128 to each pixel, then applied inverse cosine discrete transform. Then, we resized the channels according the input image, convert to color space from YCrCb to BGR and save the image to output path. Finally, print the original image size image, compressed image size, and compression ratio to terminal

In the the2_read function we get the input path, read the given image and show the user by using cv2 library.

REFERENCES

[1] Rafael C. Gonzalez and Richard E. Woods. 2006. Digital Image Processing (3rd Edition). Prentice-Hall, Inc., USA.
[2] Jpeg Compression. (2015). Arjunsreedharan.Org. https://arjunsreedharan.org/post/146070390717/jpeg-101-how-does-jpeg-work
[3] Noise reduction. (2021). Geeksforgeeks. https://www.geeksforgeeks.org/how-to-remove-noise-from-digital-image-in-frequency-domain-using-matlab/