CENG466 FUNDAMENTALS OF IMAGE PROCESSING
HW2 REPORT

Emre Zinal
Computer Engineering
Middle East Technical University
2094837

Yunus Emre Gök
Computer Engineering
Middle East Technical University
2166460

***Abstract*—The purpose of this assignment is to get familiarize with the fundamental frequency domain image enhancement techniques.**

I. Frequency Domain Filtering

Frequency domain filtering is processing the image in Frequency domain. The process consists of transforming the image with Fourier transform into frequency domain, multiplying with the filter function and finally transforming back to the spatial domain.
While removing high frequencies from image makes image smoother, removing low frequencies enhances edges.

A. Noise Reduction using Frequency Domain Filtering

In this first part our goal is to remove noise using frequency domain filtering. Noises create different patterns in the frequency domain. For the images in assignment we will now explain the noise patterns in the frequency domain.

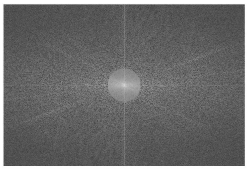

Fig. 1. Frequency domain image of A1.png

1) Butterworth Low-pass Filter: For this image (Fig. 1) we have a noise which repeats itself frequently on the spatial domain. In the frequency domain the noise has high values. We applied a Butterworth low pass filter to this image to get a resulting image without noise. Since it is a low pass filter, while removing the noise, it also made initial image smoother.
We chose Butterworth low-pass filter, because if we use an ideal filter we get a resulting image with wave-like patterns. Also we didn't chose Gaussian filter because when we applied it we got a smoother image with some details lost.

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

Fig. 2. Butterworth low-pass filter equation

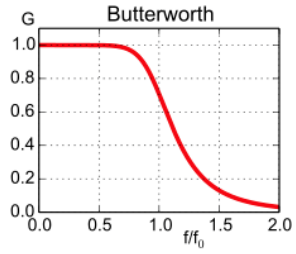

Fig. 3. Butterworth low-pass filter plot

Fig. 4. Resulting de-noised image

Here in the resulting image (Fig. 4) after we apply the Butterworth low-pass filter, we see a smoother image with initial noise patterns eliminated.

2) Bandreject Filter: Band-reject filter is a filter which passes most of the spectrum but rejects some frequencies.

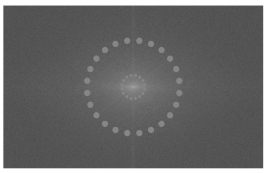

Fig. 5. Frequency domain image of A2.png

In figure 5 we see the frequency domain image of the A2.png. Here what we are seeing is two noise patterns added to the image. This patterns consists of rotated circles. We removed this patterns by designing a band-reject filter.

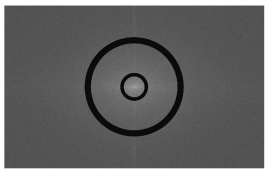

Fig. 6. The filter we designed for A2.png

Our band-reject filter consists of two rings. We adjusted their thickness to match the noise patterns'.

Fig. 7. The result of applying filter on A2.png

After we apply this filter we can clearly see the image in Fig. 7 but since we used an ideal filter we also got wave-like patterns.

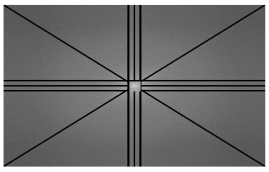

Fig. 8. The filter we designed for A3.png

3) Linear Band-reject Filter: In this part we have a color image.In spatial domain we see diagonal lines added as noise. For this part first we got the R,G and B components and examined the noise part in frequency domain. The pattern here consists of horizontal,vertical and diagonal lines. We designed an ideal filter to remove this noise pattern for all R,G, and B components. One implementation detail here was to not erase the center portion of frequency domain because it also includes important image details.

Fig. 9. The result of applying filter on A3.png

After we applied the filter we see that we removed much of the noise but there still exists some noise near the edges of the image. This might be the result of our filter not eliminate all the noise patterns.

B. Edge Detection

Edge detection is an image processing technique for finding the boundaries of objects within images. It works by detecting discontinuities in brightness. Edge detection is used for image segmentation and data extraction in areas such as image processing, computer vision, and machine vision.[1] As we see on the noise reduction edges and sudden changes like noise in part1-A3 has higher values in frequency domain. To eliminate noise we used low pass filters. In edge detection, we used high pass filters to get edges.

$$L(x,y) = \nabla^2 f(x,y) = \frac{\partial^2 f(x,y)}{\partial x^2} + \frac{\partial^2 f(x,y)}{\partial y^2}$$

Fig. 10. Laplacian Filter

In spacial domain, we used laplacian filter and using sobel filters. Laplacian in frequency domain can be obtain with: H(u,v) = -(uˆ2+vˆ2) After appling laplacian filter to image we get sudden changes in the image and nonuniform ares like details of the background in the 'B1.jpg'. To make edges hole and remove to local details like surface of the eggs or egg carton , we used gaussian filter. Gaussian filter is a low pass filter like butterworth in part1 that makes image smoother.

$$G(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{x^2}{2\sigma^2}}$$

Fig. 11. Gaussian filter

II. Image Compression with Wavelet Decomposition

Image compression is a type of data compression applied to digital images, to reduce their cost for storage or transmission.Image compression may be lossy or lossless.[3]
Lossless compression is a method used to reduce the size of a file while maintaining the same quality as before it was compressed.[4] One drawback of lossless compression is they take up much more space in comparison to lossy compression. Types of lossless compression algorithms includes raw, png, bmp.
In order to give the photo an even smaller size, lossy compression discards some parts of a photo.[4] However, the information losses can be tolerated most of the time and cannot be perceived by the end user. Lossy compression techniques achieve higher compression rates and hence used frequently in practice.
JPEG is an image compression standard that was developed by the "Joint Photographic Experts Group". JPEG is a lossy image compression method. In JPEG, It divides image to

8x8 patches and applies DCT (Discrete Cosine Transform) to these patches.[5] After DCT, Co-efficient Quantization can be applied with quantization matrix. This part is the lossy part that can't be inverted to original image. However, this process makes high frequency values to zero and human eye can't detect the difference. After quantization, lossles encoding will be applied which is Huffman Coding.
In part3 we applied another algorithm to apply a lossy compression.

A. Algorithm

1) Compress: For compression,first step we applied a Haar transform to whole image to divide it to three detail coefficients subimages. When upper left corner holds the original image, upper-right, lower left and right corners holds the vertical, horizontal and diagonal details of the image. Taking just the upper-left corner gives a good approximation for the image.
After that, we applied thresholds for the image to reduce variety of intensity and caused lost some of the information. However increasing the threshold count, can reduce the lost color values.
Until here, Some of the information can't be recovered. However, Huffman Coding make lossless encoding to store the values. MATLAB has its own Huffman functions do encode the information. To use these functions, we need PDF(probability density function). To obtain it just can just divide the histogram of the image to total number of pixels.

$$p(r_k) = \frac{h(r_k)}{M * N}$$

where $h(r_k)$ is histogram, M is row count and N is column count. After deriving pdf, we used "huffmandict" function which return dictionary for but coding and decoding information. For encoding, we vectorized the image to send it to "huffmanenco". With writing encoding, dictionary and sizes to a file, compressing part is over. MATLAB has 'save' that stores all variables in a binary format in the current directory.[6]

2) Decompress: "Huffmandeco" is for the read coded information and return it to stage before coding. This functions uses encoding and dictionary. Decoded image is in vector form after this function so we need to reshape it to make it a matrix image. Sizes saved in compression part is for this purpose. After reshaping it. We get the upper-left Haar transform. We applied inverse transformation using zero paddings on the vertical, horizontal and diagonal corners of the image. Last step is stretching histogram for the edge intensity values.

B. Result

Table I shows the difference and information loss in compression operation. We used mean square error formula here.

$$MSE = 1/n \sum (Yi - \hat{Y}i)^2$$

TABLE I
MSE VALUES

	Original vs Result	Original vs JPEG
C1	56.0461	7.6044
C2		
C3	22.1075	0.0006
C4	40.1496	4.3772
C5	20.6600	3.4384

TABLE II
COMPRESSION RATIO OF CUSTOM ALGORITHM AND JPEG FORMAT

	Bitmap(.bmp) Size/Compressed File Size	Bitmap(.bmp) Size/ JPEG File Size
C1	70.953	15.5258
C2		
C3	61.8802	14.9933
C4	38.6335	8.6619
C5	49.8635	13.8398

Mean square error function result is a estimation of quality. It is always non-negative and result closer to zero means less data loss.
As you can see in the images, there are more loss in the images with large scale of gray values. Threshold count makes this difference and increasing this count reduces the loss.
Also from table 2 we can observe that JPEG compression is superior to our compression in compression rate. We were losing much more data when try to compress more. This shows that our algorithm can be optimized more to catch up JPEG's quality.
We also observed that when we increased the compression ratio, the resulting data loss became much higher. Considering we are using lossy compression techniques and because lossy compression does not try to maintain same quality with the original image, data loss becoming higher and higher with increasing compression ratio is a relatable result. Because of that when we increase the compression rate, the resulting image had more difference than JPEG compressed image.
Different than JPEG we encoded image has a whole and this decreased the speed of decoding. Therefore, when input image has large resolution like 'C2.png', it decoding takes large amount of time and that is the reason for not having the results for C2.

REFERENCES

[1] https://homepages.inf.ed.ac.uk/rbf/HIPR2/gsmooth.htm
[2] https://www.mathworks.com/discovery/edge-detection.html
[3] https://en.wikipedia.org/wiki/Image_compression
[4] https://www.keycdn.com/support/what-is-image-compression
[5] https://www.cs.rutgers.edu/ elgammal/classes/cs334/slide9_short.pdf
[6] http://matlab.izmiran.ru/help/techdoc/ref/save.html