CENG 466 THE2

Ali Alper Yüksel
Department of Computer Engineering
METU
2036390

Ulaşcan Deniz Genç
Department of Computer Engineering
METU
2099034

I. Question 1 - Noise Reduction using Frequency Domain Filtering

There are three different images and they have different type of noises. The way of reducing the noise is using frequency domain filtering with the help of Fourier transformation.

Different type of noises need different applications of frequenct domain filterings. The solutions of noise elimination is explained separately for each picture.

A. Image A1

A1 has salt and pepper noise, it is visibly distinguishable. The noise is distributed homogeneously on the image.

We applied low-pass filter on the frequency domain of A1 with 54 pixels radius. The reason of low-pass filter is that the noise information of the image is on the out of the middle part of the frequency domain. If we take only the middle part of frequency domain, we can easily eliminate the noise without loosing the information of the image. The selection of radius is experimentally.

Frequency domain and filtered frequency domain is displayed in Figure 1.

B. Image A2

The type of noise of A2 cannot be distinguished by human eye. In frequency domain, it seems like a periodic noise. There are white holes in frequency domain of A2. These are noises and we applied band-pass filter to get rid of these holes.

We filtered frequency domain with two circles. One of them has 20 pixels thickness and other has 38 pixels thickness. The selection of thickness and distance to center was determined from the positions and sizes of white holes (e.g. noises).

Frequency domain and filtered frequency domain is displayed in Figure 2.

C. Image A3

A3 has a periodic noise. It seems like diagonal lines on the image.

We filtered frequency domain of A3 image with different type of low-pass filter. We applied diamond shape instead of circle shape to save more information of image. The dimensions of this shape was selected experimentally.

This shape was created by the using of Manhattan distance computation in algorithmic way.

Frequency domain and filtered frequency domain is displayed in Figure 3.

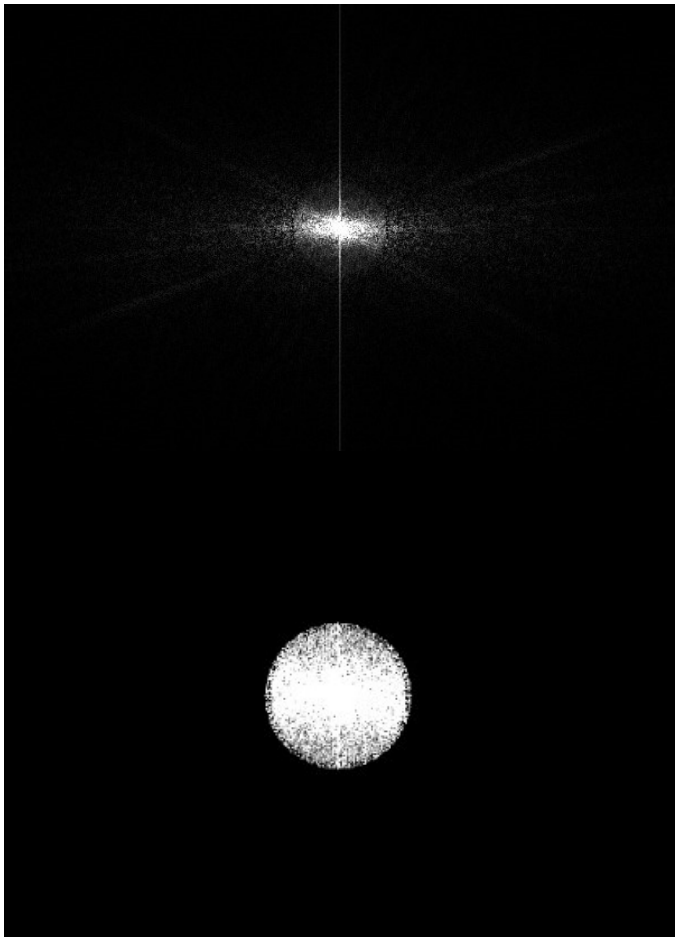

Fig. 1. FT and low-pass filtered FT of A1 image.

D. Implementation of Algorithm

We firstly took images as an input and save output as a JPG image with $imread$ and $imwrite$ functions respectively.

Then, we took Fourier transforms of images with $fft2$ function of MATLAB to get frequency domain of them. After we created filters, we shifted back filters with $fftshift$ function of MATLAB. We multiplicated filters with frequency domain of image to reduce noises. By convolution property of Fourier transform, convolution in spatial domain equals to multiplication in frequency domain.

This transformed images inversed to spatial domain with inverse Fourier transform function of MATLAB, namely $ifft2$.

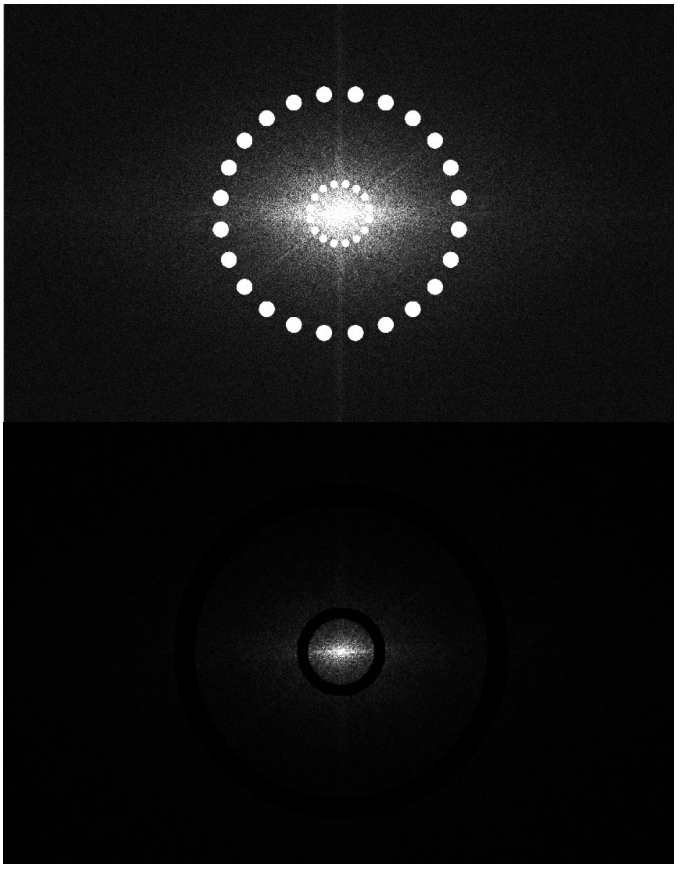

Fig. 2. FT and band-pass filtered FT of A2 image.

At this step, if we don't take absolute of inverse tranformed frequency domain, we got erroneous results at the end. If we used $ifftshift$ function of MATLAB, we didn't need to take absolute.

Finally, we normalized and mapped images into 0 and 1 to adjust image intensity values.

E. Results and Comments

Although we reduced the noise, also images corrupted a little bit since when we filtered frequency domain, we also deleted some information of image. These loss of informations made some deformation on the images.

The application of filtering frequency domain is easier than applying convolution filter. Moreover, we can save more information in frequency domain since the convolution filter must be applied on each pixel of image, which has more probability of distortion on the image.

II. Question 2 - Edge Detection

Solution of edge detection in frequency domain is using Fourier transform and high-pass filter. The reason of using high-pass filter is that the information of edges exists on out of the middle part of image in frequency domain (i.e. Fourier transformed image). If we get rid of middle part (i.e. the information of image in frequency domain), we can easily detect the edges from frequency domain.

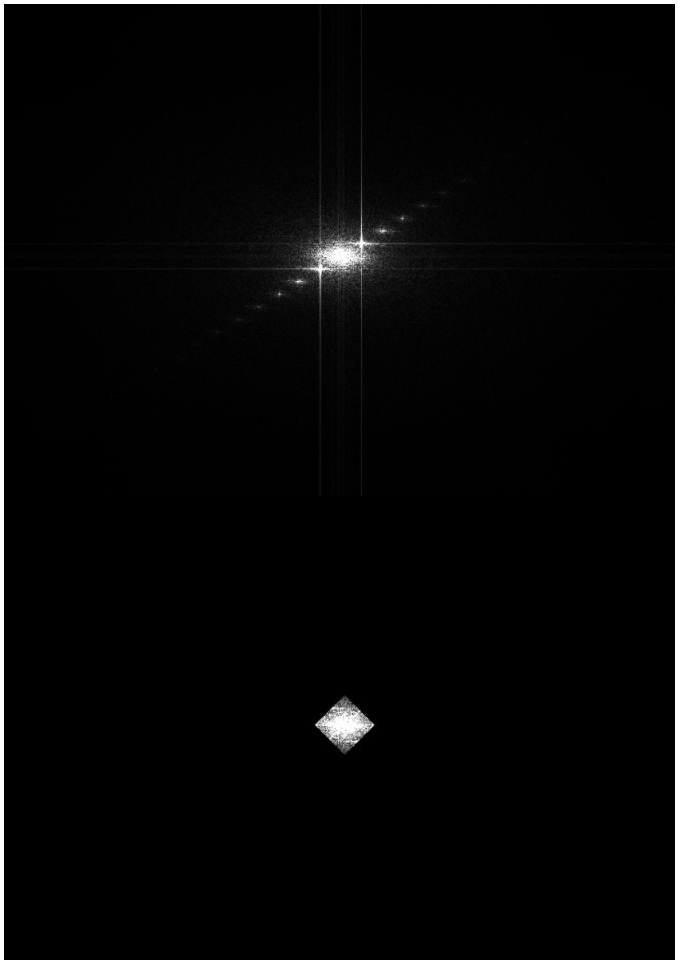

Fig. 3. FT and low-pass filtered FT of A3 image.

The selection of radius of high-pass filter depends on the amount and position of information on frequency domain. It cannot be determined with any equation. That means, we selected radius experimentally. Increasing of radius results in decreasing clearness of the edges. On the other hand, decreasing of radius results in more existence of details of image.

A. Implementation of Algorithm

We created a function that takes input image to apply edge detection. To take image as an input and save output as a JPG image, we used $imread$ and $imwrite$ functions respectively.

In function, we convert RGB image to grayscale image with $rgb2gray$ function of MATLAB in first step. This is crucial since separating image into 3 levels, applying transform to them and concatenating transformed levels didn't give any result. Also, applying edge detection into colored image gives colored edges that we didn't want.

After then, we applied Fourier transform with $fft2$ function of MATLAB. Then, we created high-pass filter with circle of radius of 50 pixels. We colored circle with black to detect the edges since most of the information that gives non-edges exist on the middle part of frequency domain.

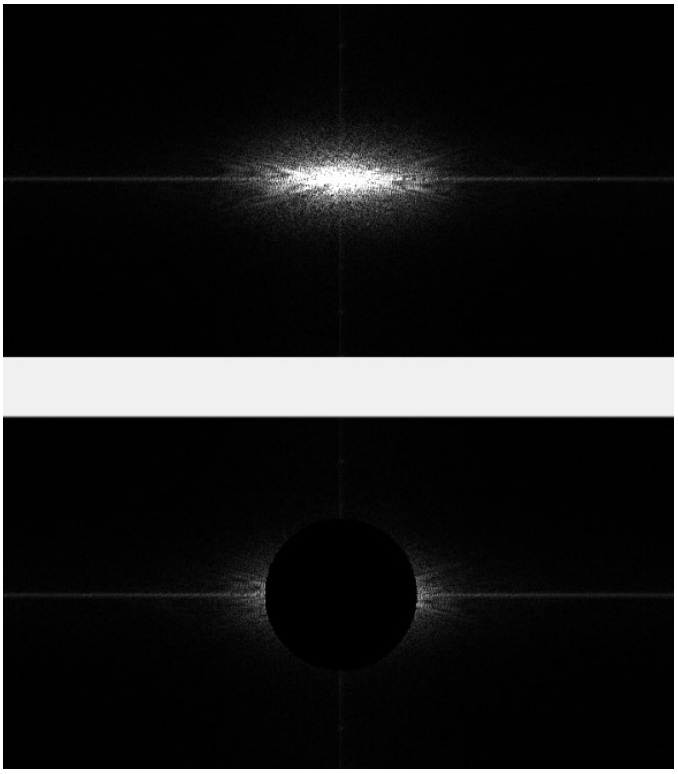

Fig. 4. FT and high-pass filtered FT of B1 image.

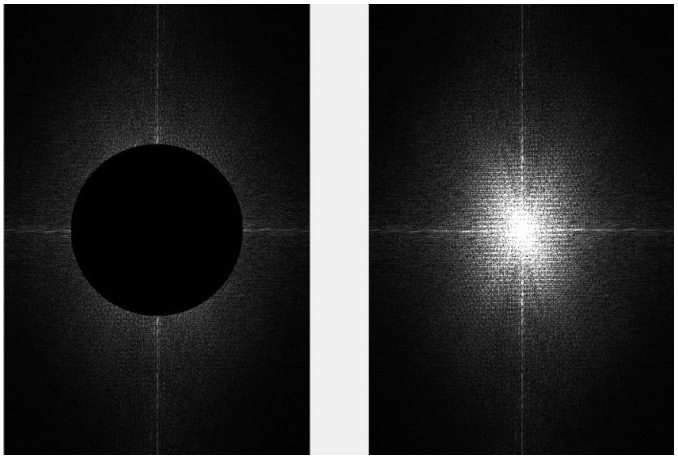

Fig. 5. FT and high-pass filtered FT of B2 image.

This filter was shifted with $fftshift$ function and multiplicated with frequency domain of image. By convolution property of Fourier transform, convolution in spatial domain equals to multiplication in frequency domain.

This transformed image inversed to spatial domain with inverse Fourier transform function of MATLAB, namely $ifft2$. Finally, we normalized and mapped this image into 0 and 1 values to adjust image intensity values.

B. Results and Comments

As can be seen on these figures, high-pass filter with 50 pixel radius produced better result. Moreover, there are some

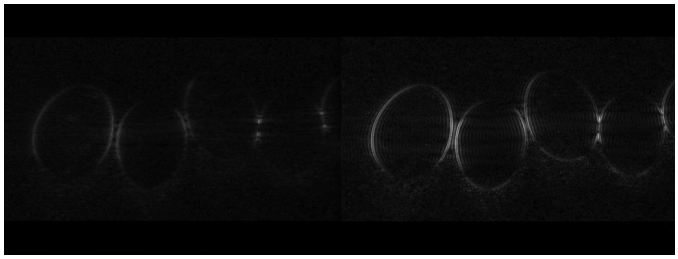

Fig. 6. Edge detection with 135 pixelled radius and 50 pixelled radius high-pass filters on B1 image.

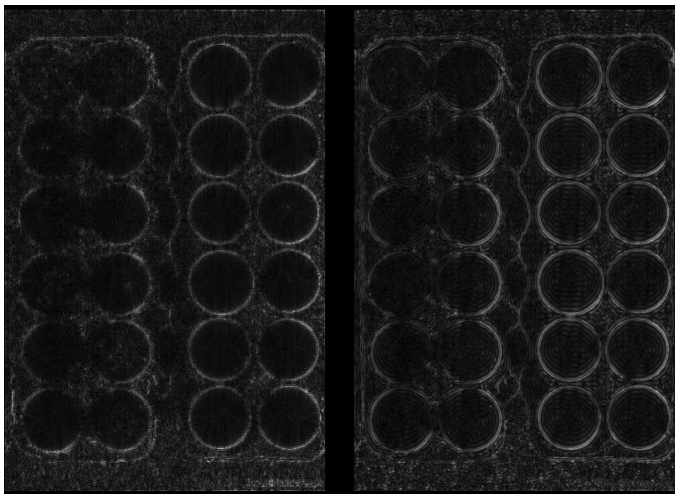

Fig. 7. Edge detection with 135 pixelled radius and 50 pixelled radius high-pass filters on B2 image.

light edges which are parallel to major edges of objects. The reason of this situation is shadows and light reflections of the objects make light edges with objects itself.

III. Question 3 - Image Compression with Wavelet Decomposition

Solution of image compression and decompression is based on wavelet transform, quantizing, encoding and decoding. When we compressed the image, losing of data is happened unfortunately.

A. Steps of the Solution and Algorithm

After we took images as inputs, we rescaled the image from integer data types to the range [0, 1] with $im2double$ function of MATLAB since $dwt2$ function needs input with double precision.

After the Haar transformation of image with double precision, we got approximated image with half of size of the image (e.g. approximated and compressed image). Then, we got thresholds with 7 levels with $multithresh$ function and added 8th threshold as the maximum value of this image. We quantized this image with these 8 threshold levels with $imquantize$ function.

Afterwards, we adjusted image into gray-scale values between 0 and 255 with $imadjust$ function. The reason of adjusting is taking histogram of this image.

Then we converted this adjusted image into array. This array and histogram would be used in computing Huffman encoding. Huffman encoding takes image as array and compresses it by using repetitive values in arrayed image.

Then we computed compression ratio and applied Huffman decoding on these compressed images with $huffmandeco$ function of MATLAB.

At the end, we reshaped compressed image into original size and took inverse Haar transform of it to get decompressed image.

B. Results and Comments

For C1 image, compression ratio is 3.206045 and mean square error is 0.003266.
For C2 image, compression ratio is 2.890494 and mean square error is 0.455598.
For C3 image, compression ratio is 2.877759 and mean square error is 0.027164.
For C4 image, compression ratio is 2.568606 and mean square error is 0.160097.
For C5 image, compression ratio is 2.922190 and mean square error is 0.154386.

As can be seen, we compressed ratio about one-third of original image and when we decompressed them, we got so many small values of MSE which means the compression and decompression algorithms work very well.

Compression and decompression in JPEG standards provides MSE's for:
- C1 is 0.000371.
- C2 is 0.000000.
- C3 is 0.000001.
- C4 is 0.003507.
- C5 is 0.010512.

Although our algorithm provides more MSE for each image, there is not very difference between them. This difference doesn't effect image very much.