CENG466 Programming Homework I

Melike Selin Aydın
1678747

Murat Türe
1746403

I. Introduction

We implemented an image processing program that does compression using several transform methods, namely, Fourier Transform, Cosine Transform and Hadamard Transform. The program is written with a graphical user interface using Matlab and Guide. User selects an input image and a transform method using the GUI and input image is compressed and written to a specific file format.

II. background

Transforms are really important in image processing as they are good as compressing energy.

Fourier Transform: Fourier transform converts equally spaced samples of a function to coefficients of complex sinuzoids. It is the transform that changes function domains from time domain to frequency domain. Being the most important transform, it is used in many practical applications such as signal processing.

Discrete Cosine Transform: It expresses data points to sum of cosine functions of different frequencies. It is not good at energy compaction as Fourier transform but it is faster. It is widely used in compression formats such as JPEG of MP3.

Hadamard Transform: Hadamard transform is another kind of transform used for energy compaction. It is the poorest at compressing among all three. It is used in formats like JPEG XR of MPEG-4 AVC.

III. methodology

A. Fourier Transform

User selects Fourier Transfrom from a listbox in the GUI and presses compress button. Compress button calls the function that runs fourier transform. After taking the transform, the function folds the transformed image corners to the center and it does compression by cutting 49% of the image from the center. For the rest of the image, pixel values are assumed to be zero. Than it takes its inverse fourier transform and writes it to a binary file.

B. Cosine Transform

User selects Cosine Transfrom from a listbox in the GUI and presses compress button. Compress button calls the function that runs discrete cosine transform using the built-in function dct2. After taking the transform, the function does compression by cutting 56% of the image from the top left corner. For the rest of the image, pixel values are assumed to be zero. Than it takes its inverse cosine transform using again a built- in Matlab function called idct2 and writes the final image to a binary file.

C. Hadamard Transform

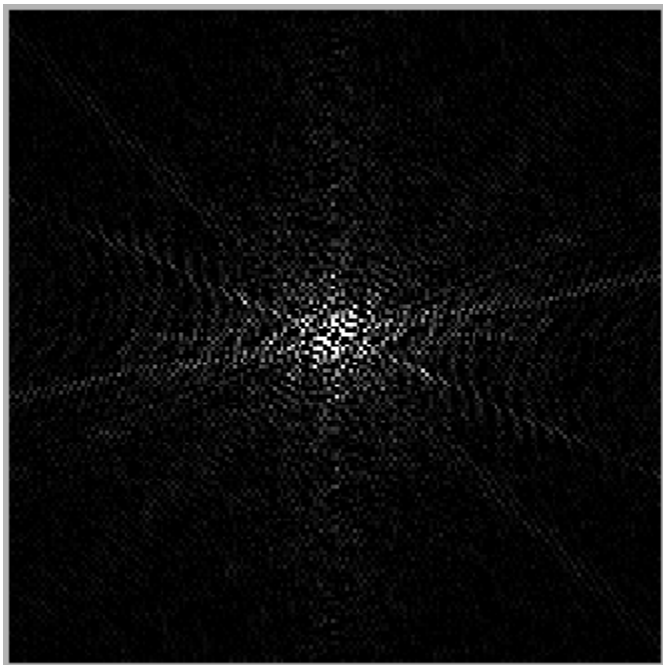
User selects Hadamard Transfrom from a listbox in the GUI and presses compress button. Compress button calls the function that runs fourier transform using the built-in function fwht. After taking the transform, the function does compression by cutting the 70% of the image from upper left corner. For the rest of the image, pixel values are assumed to be zero. Than it takes its inverse hadamard transform using again a built- in Matlab function called ifwht and writes the final image to a binary file.

IV. results

Fourier Transform gives the best outputs, when we apply Cosine Transform some information is lost. For instance, the sky appears white for the input image “cameraman.ppm”. Hadamard Transform gives results that are awkward to view. Some ghostly copies are seen all over the image.

Considering compression, Cosine Transform gives the best results. Although it compresses less than the Fourier Transform, because we write both magnitude and phase to the binary file in the case of Fourier transform and write just the magnitude in the Cosine. The files created with Hadamard transform gives the worst results since it does not compress as well as the others.

Our results can be seen below.

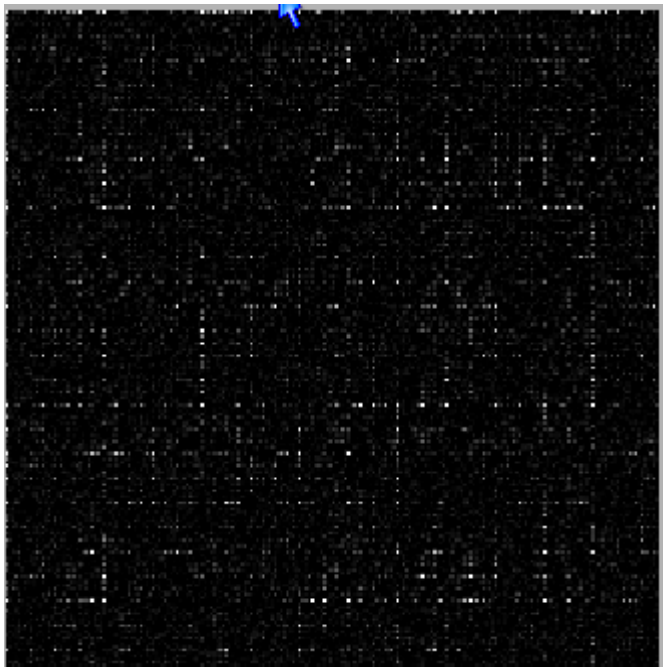


Fourier transformed image

Cosine transformed image

Image compressed with Fourier trasform

Image compressed with cosine transform



Hadamard transformed image

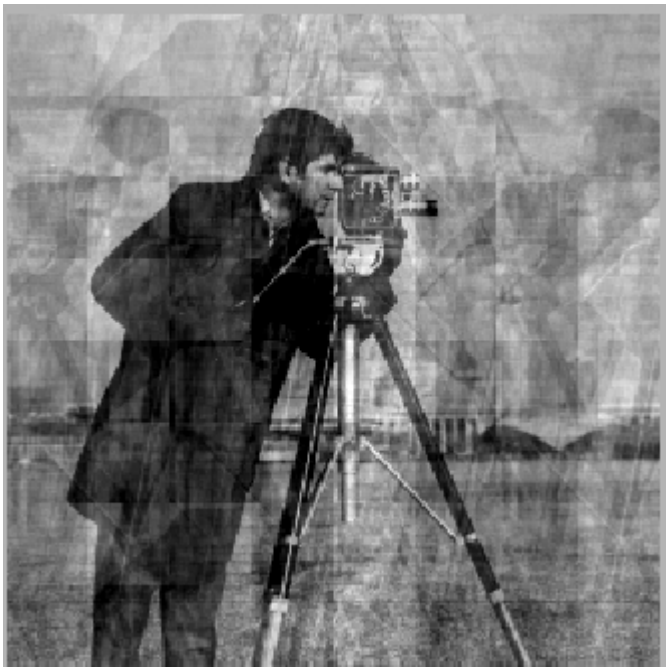


Image compressed with hadamard transform

V. WRITTEN PART

4.27 a) Spatial average = g(x, y) = (f(x, y+1) + f(x+1, y) + f(x-1, y) + f(x, y-1)) / 4

G(u, v) = ¼ (e^(2*j*pi*v/N) + e^(2*j*pi*u/M) + e^(-2*j*pi*u/M) + e^(-2*j*pi*v/N)) * F(u, v))

G(u, v) = H(u, v) * F(u, v)

So, the filter transfer function in the frequency domain is,

H(u, v) = 1/2(cos(2*pi*u/M) + cos(2*pi*v/N))

The '*' symbol is used for regular multiplication here, not for convolution.

b)In order a filter to be low pass, it should allow low frequencies and decrease as frequencies get high.
We express H(u, v) as a centered function,

H(u, v) = 1/2(cos(2*pi(v – N/2)/N) + cos(2*pi(u – M/2)/M))

When v = 0, cos(2*pi(v – N/2)/N) becomes -1.
When v = N/2, cos(2*pi(v – N/2)/N) becomes 1.
When v = N, cos(2*pi(v – N/2)/N) becomes -1, again.

So, the magnitude of the filter decreases as distance from the origin of centered filter increases. Which concludes that this is a low pass filter.

4.28 a) Spatial domain filtered function,

g(x, y) = f(x, y) – f(x+1, y) + f(x, y) – f(x, y+1)

G(u, v) = 2 * F(u, v) – F(u, v)*e^(2*pi*u/M) – F(u, v) * e^(2*pi*v/N) = (1 – e^(j*2*pi*u/M))*F(u, v) + (1 – e^(2*pi*j*v/N))*F(u, v) = H(u, v) * F(u, v)

Filter in frequency domain is

H(u, v) = -2*j(sin(pi*u/M)*e^(j*pi*u/M) + sin(pi*v/N)*e^(j*pi*v/N))

b) Similar to problem 4.27 (b)

We express H(u, v) as a centered function

H(u, v) = -2*j (sin(pi* (u-M/2)/M)*e^(j*pi*u/M) + sin(pi*(v-N/2)/N)*e^(j*pi*v/N))

When v goes from zero to M, H(u, v) goes from its peak value 2j to zero to 2j again. Therefore, this filter is a high pass filter with value zero at the origin an increases as distance from the origin increases.

4.34 If we investigate Fig. 4.41(a) we see that the only source of the periodic bright points can be the lines in the lower left corner of it.

4.36 a) If we apply highpass filter to the original image, we expect the central are to appear darker. Then it is averaged out when we apply low pass filter. Since the edges of the ring are much more sharper than the other parts of the image, it appears bright and solid after the last high pass filter.

b) The order does not matter because filtering is linear.

VI. CONCLUSION

We implemented a program with GUI that does the basic transforms namely, Fourier, Cosine and Hadamard transform. During our work we learned about transformations and how they compress the data in the image. We gained a better understanding of image compression techniques and methods used to create compressed image formats like widely used JPEG. To conclude it was a beneficial homework for us to learn about discrete transforms and basic image compression techniques.