IMAGE PROCESSING IN FREQUENCY DOMAIN USING MATLAB®: A STUDY FOR BEGINNERS

Vinay Kumar, Manas Nanda

IMAGE PROCESSING IN FREQUENCY DOMAIN USING MATLAB®: A STUDY FOR BEGINNERS

by
Vinay Kumar
and
Manas Nanda
Department of Electronics and Communication Engineering,
Jaypee University of Information Technology,
Solan-173 215, INDIA

TABLE OF CONTENTS

TITLE	PAGE
Title Page	1
Certificate ……...	2
Acknowledgement ……...	3
Table of Contents ……...	4
List of Figures ……...	6
List of Abbreviations ……...	8
Abstract ……...	9
INTRODUCTION ……...	10
Image ……...	10
Digital Image Processing ……...	10
Applications ……...	11
Image Compression ……...	12
FILTERS AND THEIR CLASSIFICATION ……...	13
Filter ……...	13
Required Classification as per Requirement ……...	13
Low Pass Filter ……...	13
High Pass Filter ……...	14
FFT Filter ……...	15
PROJECT DESCRIPTION ……...	16
Steps Involved in the Design of Filter ……...	16
Image Selection ……...	16
Matrix Representation ……...	16
Area Division ……...	17

The Working ……... 18
Phase-1 ……... 18
Phase-2 ……... 19
Phase-3 ……... 20
Phase-4 ……... 24
Phase-5 ……... 24
ADDITIONAL STUDY WORK OF IMAGE COMPRESSION USING HAAR WAVELET TRANSFORM …... 26
Wavelets ……... 26
How Does the Transformation Work ……... 26
The Compression ……... 29
LIMITATIONS OF USING MATLAB …….. 32
CONCLUSION ……... 33
BIBLIOGRAPHY ……... 34

LIST OF FIGURES

Figure 1:	Image Processing Illustration
Figure 2:	A Low-Pass Filter
Figure 3:	A high-Pass Filter
Figure 4:	MATLAB figure for a Low-Pass Filter
Figure 5:	MATLAB figure for an All-Pass Filter
Figure 6:	512x512 image of LENNA
Figure 7:	Area Division for Image Matrix
Figure 8:	Image of NOISE after calculation of FFT
Figure 9:	Sine Wave Representation
Figure 10:	FFT representation of sine wave
Figure 11:	Mesh representation of 2-D IFFT of Image
Figure 12:	3-Dimensionally Rotated version of Mesh
Figure 13:	Un-Normalized version of IFFT of Filtered Image
Figure 14:	Output Image of LENNA with Window size 10
Figure 15:	Output Image of LENNA with Window size 50

Figure 16: Output Image of LENNA with Window size 180

Figure 17: Output Image of LENNA for increasing order of size of side squares

Figure 18: Test Image to apply Haar Wavelet Transform

Figure 19: Output results for $e = 20$ and $e = 50$

LIST OF ABBREVIATIONS

FFT:	Fast Fourier Transform
IFFT:	Inverse Fast Fourier Transform
MSQE:	Mean Square Quantization Error
MATLAB:	Matrix Laboratory
ROI:	Region of Interest

CHAPTER-1
INTRODUCTION

Image

An image as defined in the "real world" is considered to be a function of two real variables, for example, a(x,y) with **a** as the amplitude (e.g. brightness) of the image at the real coordinate position (x,y).
Further, an image may be considered to contain sub-images sometimes referred to as regions-of-interest, ROIs, or simply regions. This concept reflects the fact that images frequently contain collections of objects each of which can be the basis for a region.

Digital Image Processing

Digital image processing is the use of computer algorithms to perform image processing on digital images. As a subfield of digital signal processing, digital image processing has many advantages over analog image processing; it allows a much wider range of algorithms to be applied to the input data, and can avoid problems such as the build-up of noise and signal distortion during processing.

The following picture shows what exactly an image processing does:

Figure 1

The last three pictures show red, green and blue color channels of a photograph whereas the first image is a composite.

What can be done by Image Processing ?

- Geometric transformations such as enlargement, reduction, and rotation.
- Color corrections such as brightness and contrast adjustments, quantization, or conversion to a different color space.
- Registration (or alignment) of two or more images.
- Combination of two or more images, e.g. into an average, blend, difference, or image composite.
- Interpolation and recovery of a full image from a RAW image format.
- Segmentation of the image into regions.
- Image editing and Digital retouching.
- Extending dynamic range by combining differently exposed images.

Applications

Image Processing finds applications in the following areas:

- Photography and Printing
- Satellite Image Processing
- Medical Image Processing
- Face detection, Feature detection, Face identification
- Microscope image processing

Image Compression

Image compression is the application of Data compression on digital images. In effect, the objective is to reduce redundancy of the image data in order to be able to store or transmit data in an efficient form.

Image compression can be lossy or lossless. Lossless compression is sometimes preferred for artificial images such as technical drawings, icons or comics. This is because lossy compression methods, especially when used at low bit rates, introduce compression artifacts. Lossless compression methods may also be preferred for high value content, such as medical imagery or image scans made for archival purposes. Lossy methods are especially suitable for natural images such as photos in applications where minor (sometimes imperceptible) loss of fidelity is acceptable to achieve a substantial reduction in bit rate.

Compressing an image is significantly different than compressing raw binary data. Of course, general purpose compression programs can be used to compress images, but the result is less than optimal. This is because images have certain statistical properties which can be exploited by encoders specifically designed for them. Also, some of the finer details in the image can be sacrificed for the sake of saving a little more bandwidth or storage space. This also means that lossy compression techniques can be used in this area.

The image compression technique most often used is transform coding. A typical image's energy often varies significantly throughout the image, which makes compressing it in the spatial domain difficult; however, images tend to have a compact representation in the frequency domain packed around the low frequencies, which makes compression in the frequency domain more efficient and effective. Transform coding is an image compression technique that first switches to the frequency domain, then does it's compressing. The transform coefficients should be de-correlated to reduce redundancy and to have a maximum amount of information stored in the smallest space. These coefficients are then coded as accurately as possible to not lose information.

CHAPTER-2
FILTERS AND THEIR CLASSIFICATION

Filter

A filter is a device that discriminates according to one or more attributes at its input, what passes through it.

One example is the color filter which absorbs light at certain wavelengths.

By filter design we can create filters that pass signals with frequency components in some bands, and attenuate signals with content in other frequency bands.

Required Classification as per Requirement

1. Low Pass Filter

A low-pass filter is a filter that passes low frequencies but attenuates higher than the cutoff frequency.

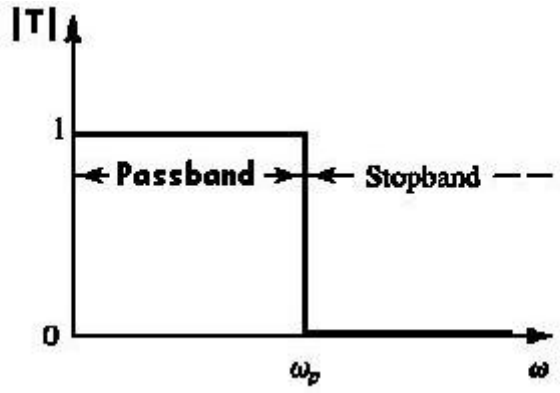


Figure 2

2. High Pass Filter

A high-pass filter is a filter that passes high frequencies well, but attenuates frequencies lower than the cut-off frequency.

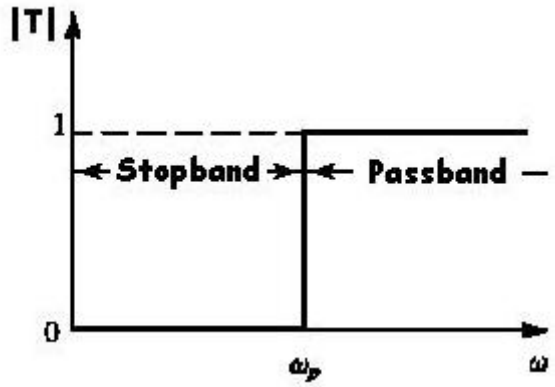


Figure 3

If we combine the above two together, we can design a filter that starts as a low-pass filter and slowly allows higher frequency components also and finally all frequencies can pass through that filter and we get the whole image.

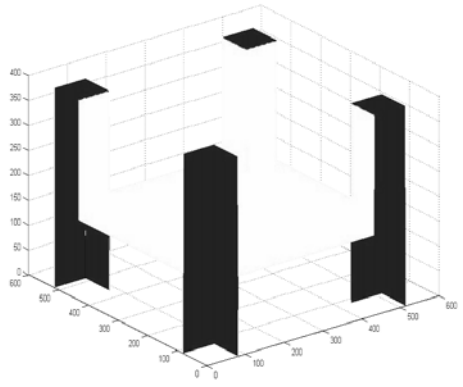


Figure 4

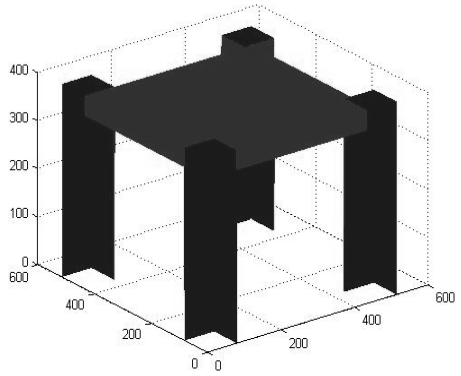
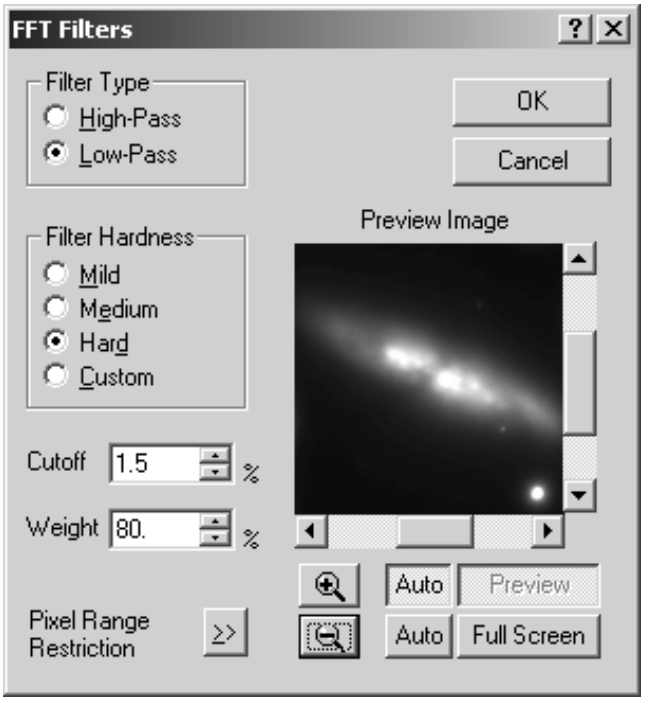


Figure 5

3. FFT Filter

FFT Filters provide precisely controlled low- and high-pass filtering (smoothing and sharpening, respectively) using a Butterworth characteristic. The image is converted into spatial frequencies using a Fast Fourier Transform, the appropriate filter is applied, and the image is converted back using an inverse FFT.



CHAPTER-3
PROJECT DESCRIPTION

Steps Involved in the Design of Filter

Image Selection

Many grey-level images are available for the purpose of showing the function and working of a filter but for standard conventions I decided to choose Lenna.

Figure 6

Matrix Representation

The image chosen is now scaled to a fixed size of (512x512) and represented as a matrix. One important thing that has to be kept in mind is that image must have its both dimensions of at least 512 or else there will be a run-time error.

Area Division

The (512x512) matrix is divided into two major portions. First of all we separate a fixed square size area (say 60x60) from all corners of the image and we assume that in any of the functions applied this area will nor be taken into account. This area will simply act as pillars of the filter. The rest of the area left comprises of the second major portion of the image matrix.

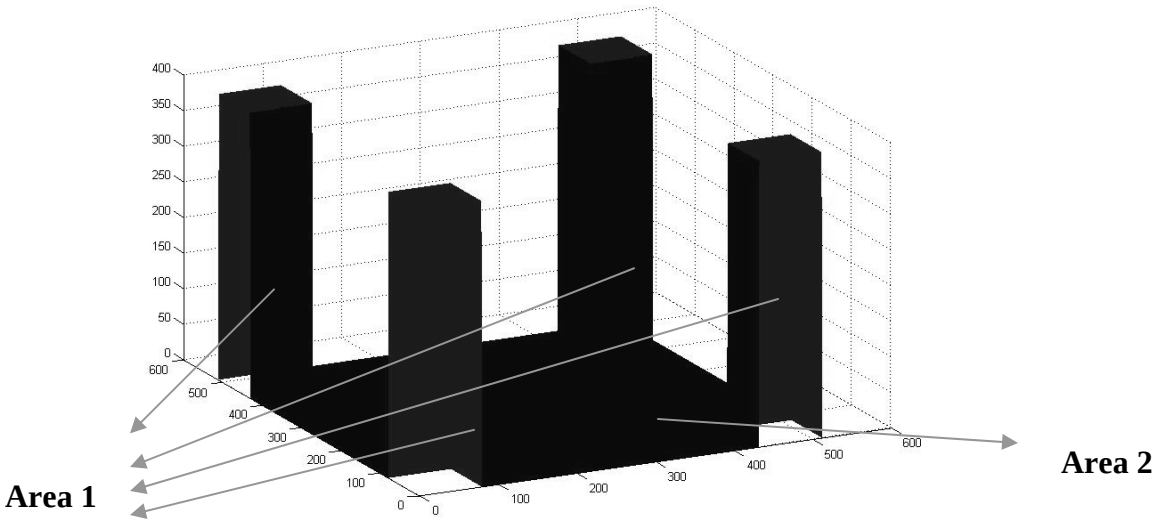


Figure 7

The Working

Phase - 1

Since MATLAB was very new to me, I started off by calculating the FFT of an image and observed some important things. When the FFT was calculated the entire image seemed to show noise all over. Since no boundary had been set for the frequencies, hence noise intrusion was very large and hence NOISE.

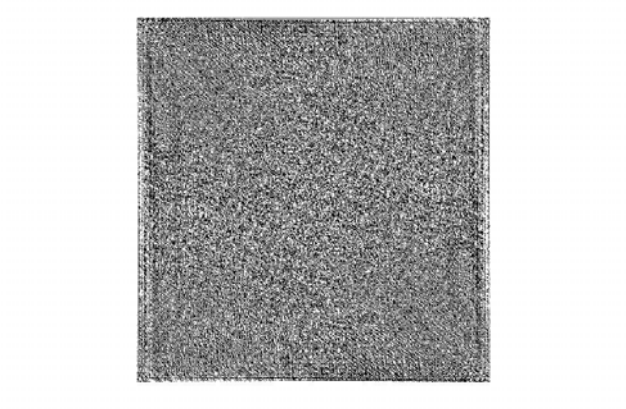


Figure 8

When 2-D FFT was calculated, proceedings made much more sense. Unless I processed a completely black image, a 2D Fourier transform of an image file (where all pixels have positive values) will always have a bright pixel in the center. That center pixel is called the DC term and represents the average brightness across the entire image.

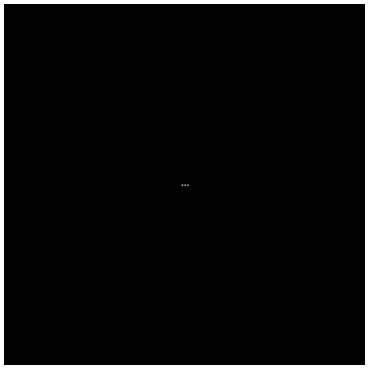


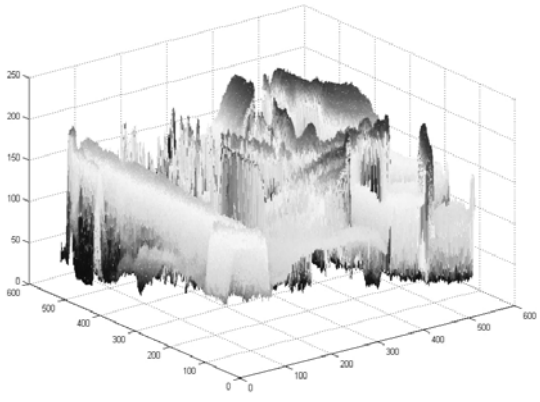
Figure 9 **Figure 10**

On the left side is an image of a sine wave where black pixels represent the bottom of the sine wave, white pixels the top and the gray pixels in between represent the sloping areas of the curve. On the right is the FFT of that image.

2D Fourier transforms are always symmetrical. The upper left quadrant is identical to the lower right quadrant and the upper right quadrant is identical to the lower left quadrant. This is a natural consequence of how Fourier transforms work.

Phase-2

When the IFFT of the image was calculated, a white screen appeared showing nothing. The reason was; since the initial 2-D FFT was nothing but noise so nothing appeared, but when the mesh/surface plot of the image was viewed, I saw the following interesting result.



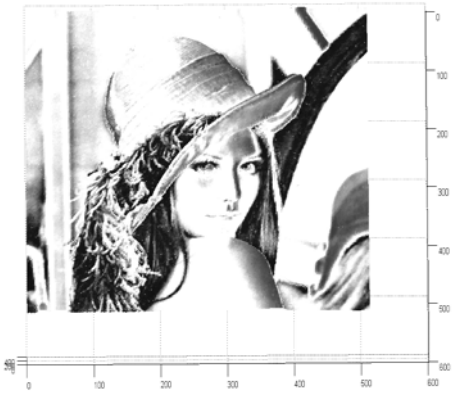


Figure11 **Figure 12**

Figure 11 clearly depicts that when the MESH function was used, the colored parametric mesh defined by the matrix arguments was visible.

On rotating it in a 3-D manner, Lenna was clearly visible as in **Figure 12**.
The reason why Lenna could not be seen was that all the values after calculating the FFT were out of the specified range i.e. they were not NORMALIZED. The moment IFFT was calculated the values were back in range but there were some differences in the original values of the image matrix and the ones obtained after IFFT.

The values were not same because in the process of NORMALIZATION round-off functions are used which might change the values at some point of time.

A few important functions that had to be used were:

RGB2GRAY : Converts RGB image to grayscale by eliminating the hue and saturation information while retaining the luminance.

IM2DOUBLE : Convert image to double precision. IM2DOUBLE takes an image as input, and returns an image of class double. If the input image is of class double, the output image is identical to it. If the input image is not double, IM2DOUBLE returns the equivalent image of class double, rescaling or offsetting the data as necessary.

Once I understood the basic concepts, I moved ahead with the filter thing. After making the side pillars, now the sheet area that was in between those pillars had to be moved up slowly so that it acts as the frequency limit till where the frequencies could enter.

Phase – 3

To move the inner sheet up so that it acts as a frequency cut-off limiter, a code was written that limited only the inner area to move and not the entire block along with the sheet.
Once this was achieved, I had to multiply the FFT of the image with this filter function that had been designed.
After multiplication the IFFT of the entire result was calculated and the filtered result was viewed for various values. The same result kept on repeating.

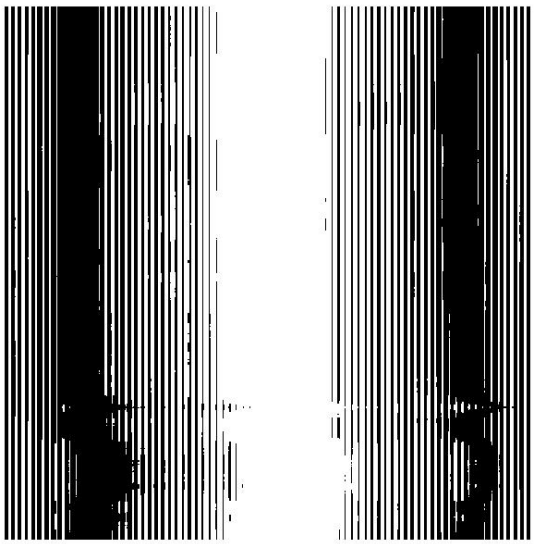


Figure 13

NORMALIZATION once again comes into picture. Since the final result had not been subjected to the specified range the result could not be viewed.

After all corrections results for various tests are displayed below:

Result-1

Filter window size 10 (Only very few low-range frequencies are allowed to pass)

Image:

Figure 14

Result-2

Filter window size 50 (A few more frequencies are allowed to pass)

Image:

Figure 15

Result-3

Filter window size 180 (A lot more frequencies are allowed to pass)

Image:

Figure 16

As you can see, by allowing the window size to increase and bringing the resultant frequencies within the range, the resulting filtered image becomes more and more clear.

Phase-4

Calculation of MSQE

Mean square quantization error (MSQE) is a figure of merit for the process of analog to digital conversion.
As the input is varied, the input's value is recorded when the digital output changes. For each digital output, the input's difference from ideal is normalized to the value of the least significant bit, then squared, summed, and normalized to the number of samples.

MSQE calculations were carried out for all the results abtained after filtering and were within permissible range i.e **-3.6% to +3.6%** of the original value.

Phase-5

Size of Outer Squares

The area that comprised of the four outer squares also plays a very important part in the final outlook of the image. Larger the square size, sharper is the final image.
The reason of such an outcome is that when the squares are chosen, the particular area is not being operated by any of the functions. Whenever a function acts on the matrix the image covered under the four squares is untouched and in the final output appears as it is.
Hence there is a major change in the final image as we increase or decrease the size of the squares.

Observation : Larger is the size of the outer squares, better is the final image.

For an example, a few of the images have been shown below in increasing order of size of squares.

Figure-17

CHAPTER-4

ADDITIONAL STUDY WORK OF IMAGE COMPRESSION USING HAAR WAVELET TRANSFORM

Wavelets

Wavelets provide a mathematical way of encoding numerical information (data) in such a way that it is layered according to level of detail. This layering not only facilitates the progressive data transmission mentioned above, but also approximations at various intermediate stages. The point is that these approximations can be stored using a lot less space than the original data, and in situations where space is tight, this data compression is well worthwhile.

How Does The Transformation Work

We describe a scheme for transforming large arrays of numbers into arrays that can be stored and transmitted more efficiently; the original images (or good approximations of them) can then be reconstructed by a computer with relatively little effort. For simplicity, we first consider the 8x8 = 64 pixel image in the figure below, which is a blow up of a region around the nose. The region extracted is blacked as shown.

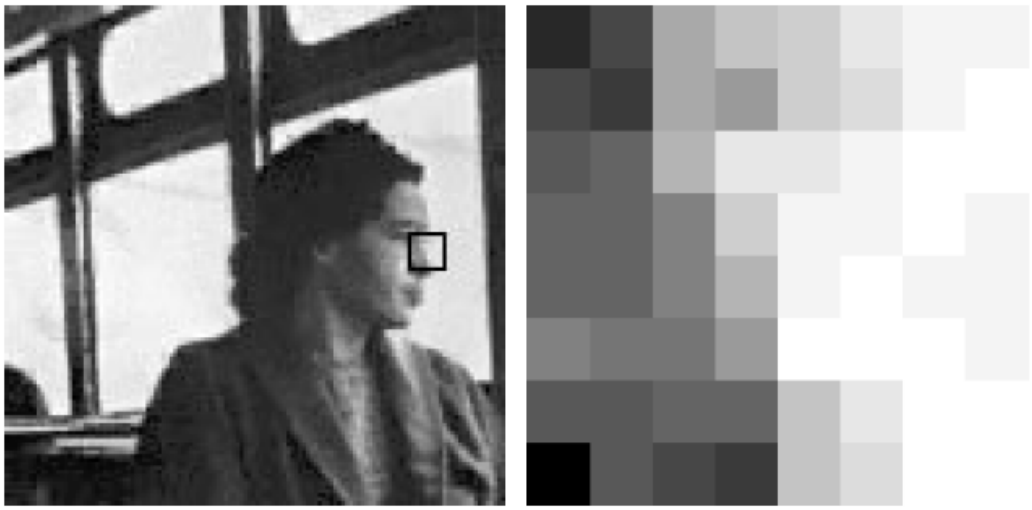


Figure-18

This image is represented by rows 60 to 67 and columns 105 to 112 of the matrix. We now display and name this sub-matrix:

$$P = \begin{pmatrix} 576 & 704 & 1152 & 1280 & 1344 & 1472 & 1536 & 1536 \\ 704 & 640 & 1156 & 1088 & 1344 & 1408 & 1536 & 1600 \\ 768 & 832 & 1216 & 1472 & 1472 & 1536 & 1600 & 1600 \\ 832 & 832 & 960 & 1344 & 1536 & 1536 & 1600 & 1536 \\ 832 & 832 & 960 & 1216 & 1536 & 1600 & 1536 & 1536 \\ 960 & 896 & 896 & 1088 & 1600 & 1600 & 1600 & 1536 \\ 768 & 768 & 832 & 832 & 1280 & 1472 & 1600 & 1600 \\ 448 & 768 & 704 & 640 & 1280 & 1408 & 1600 & 1600 \end{pmatrix}.$$

To demonstrate how to wavelet transform such a matrix, we first describe a method for transforming strings of data, called **Averaging and Differencing**.

Afterwards, we'll use this technique to transform an entire matrix as follows:

- Treat each row as a string, and perform the averaging and differencing on each one to obtain a new matrix, and then apply exactly the same steps on each column of this new matrix, finally obtaining a row and column transformed matrix.
- To understand what averaging and differencing does to a data string, for instance the 1st row in the matrix P above, consider the table below. Successive rows of the table show the starting, intermediate, and final results.

576	704	1152	1280	1344	1472	1536	1536
640	1216	1408	1536	**-64**	**-64**	**-64**	**0**
928	1472	**-288**	**-64**	**-64**	**-64**	**-64**	**0**
1200	**-272**	**-288**	**-64**	**-64**	**-64**	**-64**	**0**

There are 3 steps in the transform process because the data string has length **8 = 2^3**. The first row in the table is our original data string, which we can think of as four pairs of numbers. The first four numbers in the second row are the averages of those pairs. Similarly, the first two numbers in the third row are the averages of those four averages, taken two at a time, and the first entry in the fourth and last row is the average of the preceding two computed averages.

The remaining numbers, shown in **bold**, measure deviations from the various averages. The first four bold entries, in the second half of the second row, are the result of subtracting the first four averages from the first elements of the pairs that gave rise to them: subtracting **640;1216;1408;1536** from **576,1152,1344,1536**, element by element, yields **-64,-64,-64,0.** These are called detail coefficients; they are repeated in each subsequent row of the table. The third and fourth entries in the third row are obtained by subtracting the first and second entries in that row from the first elements of the pairs that start row two: subtracting **928;1472** from **640;1408**, element by element, yields **-288,-64**. These two new detail coefficients are also repeated in each subsequent row of the table. Finally, the second entry in the last row, **-272**, is the detail coefficient obtained by subtracting the overall average, **1200**, from the **928** that starts row three.

Observation:

I have transformed my original string into a new string in 3 steps. Moreover, the averaging and differencing process is reversible: we can work back from any row in the table to the previous row and hence to the first row by means of appropriate additions and subtractions. In other words, we have lost nothing by transforming our string.

To apply the scheme to an 8x8 matrix, we simply do the averaging and differencing three times on each row separately, and then three times on the columns of the resulting matrix. Averaging and differencing columns can also be achieved by transposing the row-transformed matrix, doing row transformations to the result of that transposition, and transposing back. The final result is a new 8x8 matrix **T**, called the **Haar Wavelet Transform** of **P**.
Applying this technique to the matrix P as above, we obtain, after a great deal of calculation, the transformed matrix **T.**

$$T = \begin{pmatrix} 1212 & -306 & -146 & -54 & -24 & -68 & -40 & 4 \\ 30 & 36 & -90 & -2 & 8 & -20 & 8 & -4 \\ -50 & -10 & -20 & -24 & 0 & 72 & -16 & -16 \\ 82 & 38 & -24 & 68 & 48 & -64 & 32 & 8 \\ 8 & 8 & -32 & 16 & -48 & -48 & -16 & 16 \\ 20 & 20 & -56 & -16 & -16 & 32 & -16 & -16 \\ -8 & 8 & -48 & 0 & -16 & -16 & -16 & -16 \\ 44 & 36 & 0 & 8 & 80 & -16 & -16 & 0 \end{pmatrix}.$$

This matrix has one overall average value in the top left hand corner, and **63** detail elements. The first row is not the same as the last row in the table we saw before, since this time, column as well as row transformations have been done.

The **Point** of the **Wavelet Transform** is that *regions of little variation in the original data manifest themselves as small or zero elements in the wavelet transformed version.*

The 0's in T are due to the occurrences of identical adjacent elements in P, and the **-2, -4**, and **4** in **T** are result of some of the nearly identical adjacent elements in P.

The Compression

The real pay-off in the wavelet transmission game is not so much the expectation of sparsity of the transformed matrices, it's the fact that we can fiddle with the “mostly detail" versions to make lots of entries zero: we can alter the transformed matrices, taking advantage of “regions of low activity" and then apply the **Inverse Wavelet Transform** to this doctored version, to obtain an approximation of the original data.

Thus we arrive at the door of wavelet compression:

Fix a nonnegative threshold value $\boldsymbol{e}$, and decree that any detail coefficient in the wavelet transformed data whose magnitude is less than or equal to $\boldsymbol{e}$ will be reset to zero (hopefully, this leads to a relatively sparse matrix), then rebuild an approximation of the original data using this doctored version of the wavelet transformed data. The surprise is that in the case of image data, we can throw out a sizable proportion of the detail coefficients in this way and obtain visually acceptable results. This process is called **Lossless Compression** when no information is lost (e.g., if $\boldsymbol{e} = 0$); otherwise it's referred

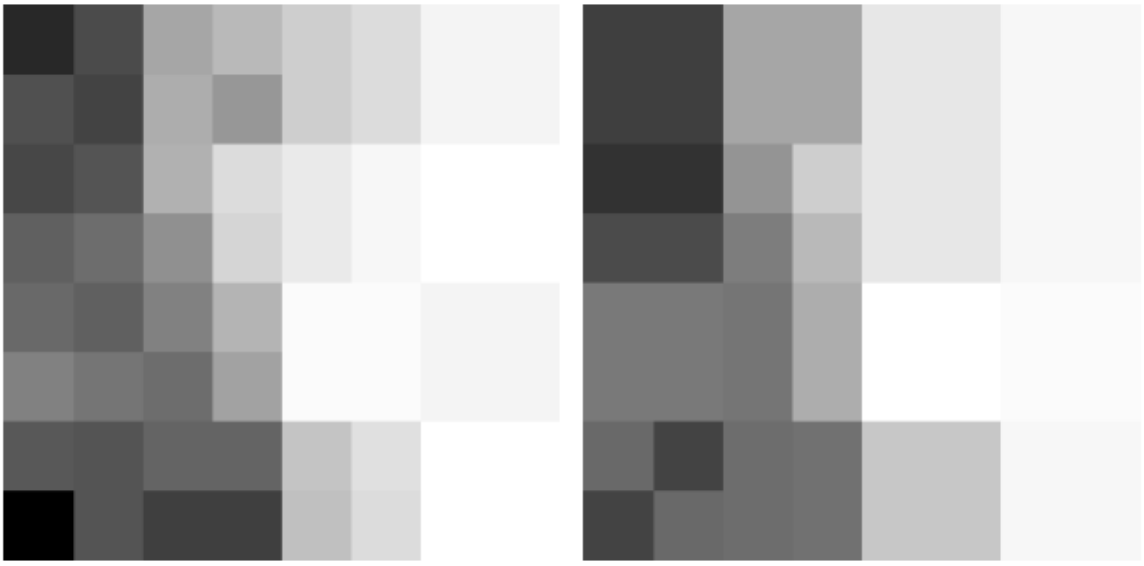
to as lossy compression (in which case $\boldsymbol{e} > 0$). In the former case we can get our original data back, and in the latter we can build an approximation of it.

For instance, consider the 8x 8 image matrix **P** and its wavelet transformed version **T** from before. If we take $\boldsymbol{e} = 20$, i.e., reset to zero all elements of **T** which are less than or equal to **20** in absolute value, we obtain the doctored matrix:

$$D = \begin{pmatrix} 1212 & -306 & -146 & -54 & -24 & -68 & -40 & 0 \\ 30 & 36 & -90 & 0 & 0 & 0 & 0 & 0 \\ -50 & 0 & 0 & -24 & 0 & 72 & 0 & 0 \\ 82 & 38 & -24 & 68 & 48 & -64 & 32 & 0 \\ 0 & 0 & -32 & 0 & -48 & -48 & 0 & 0 \\ 0 & 0 & -56 & 0 & 0 & 32 & 0 & 0 \\ 0 & 0 & -48 & 0 & 0 & 0 & 0 & 0 \\ 44 & 36 & 0 & 0 & 80 & 0 & 0 & 0 \end{pmatrix}.$$

Applying the **Inverse Wavelet Transform** to **D**, we get the reconstructed approximation

$$R = \begin{pmatrix} 582 & 726 & 1146 & 1234 & 1344 & 1424 & 1540 & 1540 \\ 742 & 694 & 1178 & 1074 & 1344 & 1424 & 1540 & 1540 \\ 706 & 754 & 1206 & 1422 & 1492 & 1572 & 1592 & 1592 \\ 818 & 866 & 1030 & 1374 & 1492 & 1572 & 1592 & 1592 \\ 856 & 808 & 956 & 1220 & 1574 & 1590 & 1554 & 1554 \\ 952 & 904 & 860 & 1124 & 1574 & 1590 & 1554 & 1554 \\ 776 & 760 & 826 & 836 & 1294 & 1438 & 1610 & 1610 \\ 456 & 760 & 668 & 676 & 1278 & 1422 & 1594 & 1594 \end{pmatrix}.$$



***e* = 20** ***e* = 50**

Figure-19

The first figure shows the image corresponding to **R**; compare this with cropped nose figure. The second figure shows what happens if we repeat with ***e*** = 50; this clearly deviates even further from the original image. Admittedly, these approximations are not visually impressive, but this is mainly because of the scale; similar approximations of much higher resolution images (more pixels) are often quite acceptable.

Thus, I've shown how we can compress an image using **Haar Wavelet Transform.**
I tried to work on the cropped portion but when the solution is used for images of much higher resolution, the results obtained are excellent.

CHAPTER-5
LIMITATIONS OF USING MATLAB

- Because MATLAB is a proprietary product of The MathWorks, users are subject to vendor lock-in.

- MATLAB lacks a package system, like those found in modern languages such as Java and Python, where classes can be resolved unambiguously. In MATLAB, all functions share the global namespace, and precedence of functions with the same name is determined by the order in which they appear in the user's MATLAB path and other subtle rules. As such, two users may experience different results when executing what otherwise appears to be the same code when their paths are different.

- Many functions have a different behavior with matrix and vector arguments. Since vectors are matrices of one row or one column, this can give unexpected results.

- Though other datatypes are available, the default is a matrix of doubles. This array type does not include a way to attach attributes such as engineering units or sampling rates.

CHAPTER-6
CONCLUSION

There are numerous other filters for the Image Compression process, just like the one I have designed. So what makes my project different from others ?

The basic concept underlying in my Filter Design is that using simple 2-D FFT and IFFT approach we can restrict the frequencies according to our choice. It depends entirely on the user to mould the filter specifications and allow a particular range of frequencies from the origin to pass through and observe the result obtained as an image.

Also the side squares approach helps us to find a solution where minor (sometimes imperceptible) loss of fidelity is acceptable to achieve a substantial reduction in bit rate and some of the finer details in the image can be sacrificed for the sake of saving a little more bandwidth or storage space.

Currently, design of filters with a very high precision and degree of control are not available. I hope that my effort is going to find applications in near future.

BIBLIOGRAPHY

[1] Gonzalez and Woods, *Digital Image Processing*. Pearson Education Inc., 2002

[2] P.Ramesh Babu, *Digital Image Processing*. Scitech Publications., 2003

[3] Rudra Pratap, *Getting Started With MATLAB 7*. Oxford University Press, 2006

[4] Bracewell, R.N. , *Two Dimensional Imaging*. Prentice Hall, 1995

[5] www.ieeexplore.com

[6] www.mathwork.com

[7] www.wikipedia.com