Linear Algebra Project

MAT204

Faculty - Gaurav Goswami

Group-1

Project Title: Image Compression using Haar Wavelet Based Approach

Group Members

Name	Enrollment Number
Dev Jani	AU1940045
Yashvi Navadia	AU1940123
Vishwa Raval	AU1940131
Darsh Patel	AU1940150

1. Introduction:

Background

Nowadays the data transmission takes place in the form of images, graphics, audio and video. These types of informative data require a lot of storage capacity and transmission bandwidth. Consequently, the theory of data compression becomes more significant for reducing the data redundancy in order to conserve more transfer energy and save storage of data. In this context, this project addresses the problem of the compression of images. There are numerous algorithms available for data compression such as Huffman Encoding, Lempel Ziv Encoding for lossless image compression, and also transform coding methods such as Fourier Transform, Discrete Cosine Transform, and Wavelet Transform for lossy image compression. However, in this study **we aim to address the problem of a low complex 2D Image Compression using the Haar Wavelets as the basis functions. We also intend to calculate the quality of compressed images** using Peak Signal to Noise Ratio (PSNR) factor [1].

Motivation and Overview

Wavelets are able to yield a mathematical method for programming numerical data in a manner that it is layered in detail, level-wise. Haar Wavelet is one such kind of wavelet. The layering can simplify approximations at intermediate stages, leading to less space for data storage than the original required space [2]. We have described the details of linear algebra concepts incorporated with the Haar Wavelet Transform in the following part of the report. Further, we have used MATLAB software to perform all the computations and visualisation mentioned in this work. The original goal was to be able to compress any types of stationary images using the Haar. However, currently our approach works only for greyscale images, and not for the images containing RGB components.

2. Approach:

A detailed description of your approach

As mentioned in the Introduction, we will be using Greyscale Images as the part of input because as mentioned in [3], that the pixels (collection of discrete small cells for example rectangular or square) of greyscale images has pixel values which are single number that represents the pixel brightness level. These are stored as 8-bit integers giving us the values from 0 to 255, where typically 255 represents the white, 0 is black and the intermediate values are different shades of gray [3]. **Reason for using Greyscale images as input in Haar wavelet Transform** is that the colour gray colour has one, whereas the RGB components has 3 colours [4], so rather than dealing with 3 intensities at a time it is preferred to specify a single intensity for computational advantages. As mentioned in [4] and [5], image processing of Greyscale images are preferred over colour images (or in many cases converted from RGB to gray) is mainly due to the storage issue where RGB components require 24 bits (8 each) and Greyscale only 8 bit, as well as it is easier and convenient to deal with single layered image rather than a three layered image for a variety of tasks (in our case, image compression).

An overview to Down-sampling/ Decimation

A two-dimensional image with higher resolution can be divided into little (2×2) blocks of pixels, and each block is replaced by a pixel with intensity equal to the average of the intensities of the pixels it replaced. This process is known as Down-sampling or Decimation [6]. An (n x n) image, (where n is a power of two) can be down-sampled to obtain an (n/2 × n/2) image.

A simple example of down-sampling of 16-pixel one dimensional image:

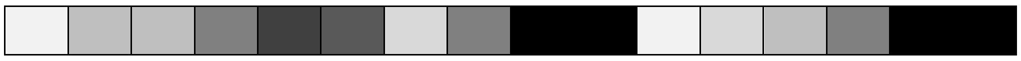


Image 1:1-D image with 16 pixels

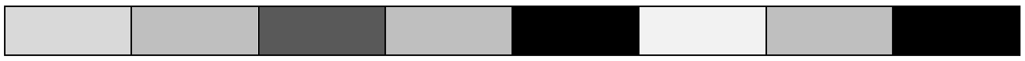


Image 2:1-D image with 8 pixels (got from down-sampling of Image-1)
In the same way we can further down-sample this image to (1 x 1) image.

Further, the detailed explanation of the process is described through mathematical operations in the next section.

Linear Algebra Concepts Used:

List of basic LA concepts used:

1. Basis of a vector space (Standard basis, Orthogonal basis)
2. Vector subspace
3. Change of basis
4. Orthogonal complements
5. Orthogonal projection
6. Orthogonalization of a set of vectors in the basis
7. Orthonormal basis
8. Inner Product Space

Building on the **fundamentals of one-dimensional image compression**, which is less complicated, we understood the analogy of the Linear Algebra concepts explained for two dimensional images through one dimensional image.

Haar wavelets for one-dimensional images: [6]

Deriving Haar Basis for a vector space V_n:

- STEP1: Deriving box basis for $\mathbf{V}_n$, $\mathbf{V}_{n/2}$… $\mathbf{V}_1$

Logically, an n-pixel image can be considered as an n-vector. Now, linear algebra helps in visualizing the vectors of images in a single vector space. Different images can be represented in terms of different bases for the same vector space. For the purpose of compression, it is advantageous to use **Orthonormal bases**.

The number of pixels in the image with highest resolution, n, is assumed to be a power of two, which makes the method less complicated.

⇨ Let, n = 16,

Here the vector space we will be dealing with is $\mathbb{R}^{16}$, which is denoted as $\mathbf{V}_{16}$ in this section.

We choose standard basis as an orthogonal basis of $\mathbf{V}_{16}$.It is also known as the "Box basis" in this context.

The set of box basis = {$\mathbf{b}_0$, $\mathbf{b}_1$ … $\mathbf{b}_{15}$}

Where, $\mathbf{b}_0$ =

1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

$\mathbf{b}_1$ =

0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0

.
.
.

$\mathbf{b}_{15}$ =

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

So, any one dimensional 16-pixel image can be represented as a linear combination of the vectors $\mathbf{b}_0$, $\mathbf{b}_1$ … $\mathbf{b}_{15}$.

For example, Image-1, given by a vector **v,**

Where, **v** =

2	4	4	5	9	7	3	5	0	0	2	3	4	5	0	0

can be represented as $\mathbf{v} = 2\mathbf{b}_0 + 4\mathbf{b}_1 + \ldots 0\mathbf{b}_{15}$.

⇨ Now, in order represent 8-pixel images(which are derived from down sampling of 16-pixel images) as vectors in $\mathbb{R}^{16}$, we need to define $\mathbf{V}_8$ in a manner such that its "box vectors" (vectors in the box basis) $\mathbf{b'}_0$, $\mathbf{b'}_1$, … $\mathbf{b'}_7$ are

$\mathbf{b'}_0$ =

1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0

$\mathbf{b'}_1$ =

0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0

.
.
.

$\mathbf{b'}_7$ =

0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1

⇨ Similarly, we can get basis for $\mathbf{V}_4$, $\mathbf{V}_2$, and $\mathbf{V}_1$. Box basis of any of $\mathbf{V}_i$, for i = 1…n will be an **Orthogonal basis.**

- STEP2: Consider **orthogonal complements** of low-resolution images in vector space of high-resolution images.

The vector space generated by this method is known as **wavelet space**. If the wavelet space W_i is orthogonal complement of $\mathbf{V}_i$ in $\mathbf{V}_{2i}$ then from the **properties of orthogonal complements** we can say that **the basis of V_{2i} is a union of bases of W_i and V_i.**

From this property, we can calculate that

One basis for V_n is the union of:

A basis for V_1, a basis for W_1, a basis for W_2, a basis for W_4…a basis for $W_{n/2}$. Such a basis is called Haar Basis, and the vectors forming this basis are called wavelet vectors.

<u>Finding vectors in the basis set of a wavelet space (W_i):</u>

<u>General Case [6]:</u>

<u>Claim</u>: For every x<n, where x and n both are power of 2, the squared norm of basis vectors of W_k, $\mathbf{wx}_i$, where i=0 … (k-1) is **n/4x**.

We will explain this general case with the help of previous example where n=16,

In this case we first get W_8, the orthogonal complement of $\mathbf{V}_8$ in $\mathbf{V}_{16}$. Here, our aim is to find the vectors $\mathbf{w8}_0$, $\mathbf{w8}_1$, …$\mathbf{w8}_7$ i.e. the vectors in the basis of W_8.

Using the general method for Orthogonal Projection in $\mathbb{R}^n$, (the one covered in the class) when we find the orthogonal projection of vectors in the box basis of $\mathbf{V}_{16}$ on the basis of V_8, we get eight zero vectors and other eight vectors which form basis for W_8, that are as follows:

$\mathbf{w8}_0$=

0.5	-0.5	0	0	0	0	0	0	0	0	0	0	0	0	0	0

$\mathbf{w8}_1$ =

0	0	0.5	-0.5	0	0	0	0	0	0	0	0	0	0	0	0

…

$\mathbf{w8}_7$ =

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.5	-0.5

Squared norm of each of these vectors = 0.5

The exact same approach for W_4 gives:

$\mathbf{w4}_0$ =

0.5	0.5	-0.5	-0.5	0	0	0	0	0	0	0	0	0	0	0	0

$\mathbf{w4}_1$ =

0	0	0	0	0.5	0.5	-0.5	-0.5	0	0	0	0	0	0	0	0

$\mathbf{w4}_2$ =

0	0	0	0	0	0	0	0	0.5	0.5	-0.5	-0.5	0	0	0	0

w4$_3$ =

0	0	0	0	0	0	0	0	0	0	0	0	0.5	0.5	-0.5	-0.5

Squared norm of each of these vectors = 1

Similar process with W$_2$ as well as W$_1$ gives:

w2$_0$ =

0.5	0.5	0.5	0.5	-0.5	-0.5	-0.5	-0.5	0	0	0	0	0	0	0	0

w2$_1$ =

0	0	0	0	0	0	0	0	0.5	0.5	0.5	0.5	-0.5	-0.5	-0.5	-0.5

Squared norm of these vectors = 2

0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	-0.5	-0.5	-0.5	-0.5	-0.5	-0.5	-0.5	-0.5

and **w1**$_0$ =

Squared norm of these vectors = 4

These above mentioned 15 vectors and in addition to that the basis vector of **V**$_1$, whose norm is n=16

1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Form Haar wavelet basis for **V**$_{16.}$

Mathematics behind decomposition using wavelet transforms

INPUT: List of intensities of original image. i.e. **v** = [2 4 4 5 9 7 3 5 0 0 2 3 4 5 0 0]
OUTPUT: (n-1) wavelet coefficients (using differencing), plus one overall average intensity.

(Explained through decomposition of V$_{16}$)

Initially the box basis was the basis set for **V**$_{16}$. The initial stage of wavelet transformation decomposes**V**$_{16}$ into **V**$_8$ and its orthogonal complement W$_8$. That gives a new set of a basis: a union of the box basis of **V**$_8$ and wavelet basis for W$_8$. (Basic concept: Change of basis)

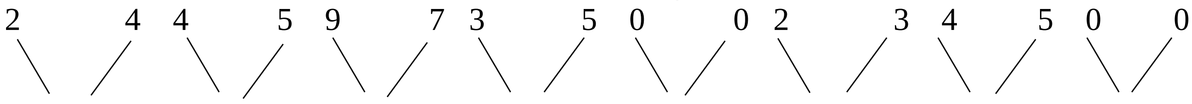
As computed while deriving Haar Basis for a vector space, the vector representation of Image 1 is **v** = 2**b**$_0$ + 4**b**$_1$ + … 0**b**$_{15.}$ We aim to represent this vector as a linear combination of **b'**$_0$, **b'**$_1$ … **b'**$_7$ and **w8**$_0$, **w8**$_1$ …w87.

Since the vectors of linear combination are mutually orthogonal, we can find the coefficients found using the following formula formulae: (Basic concept used: **Inner Product**)

Coefficient of $\mathbf{b'}_i = (\mathbf{v} \cdot \mathbf{b'}i)/ (\mathbf{b'}_i. \mathbf{b'}i)$ -------------------- (1)
Coefficient of $\mathbf{w8}_i = (\mathbf{v} \cdot \mathbf{w8}i)/ (\mathbf{w8}_i. \mathbf{w8}i)$ ----------------- (2)

1. <u>Averaging</u>:

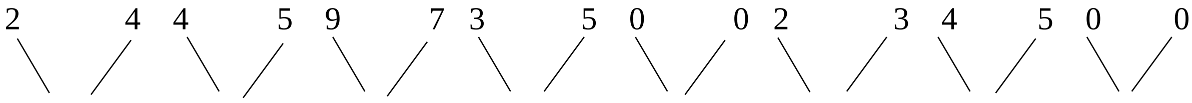
Calculation using formula-1 gives the coefficient of $\mathbf{b'}_i$to be equal to the average of entries at the indices 2i and 2i + 1 in **v**. For i = 0, 1, 2... 7. This method is famously known as "**Averaging**" in the wavelet theory.
Therefore, first part of the coordinate vector showing the coefficients of$\mathbf{b'}_1$ … $\mathbf{b'}_7$is:



3	9/2	8	4	0	5/2	9/2	0

2. <u>Differencing</u>:

Calculation using formula-2 gives the coefficient of $\mathbf{w8}_i$to be equal to the difference of entries at the indices 2i and 2i + 1 in **v**. For i = 0, 1, 2... 7. This method is famously known as "**Differencing**" in the wavelet theory.
So, the second part of the coordinate vector showing the coefficients of $\mathbf{w8}_0$, $\mathbf{w8}_1$ …w87 is:



-2	-1	2	-2	0	-1	-1	0

From the above two sections, we can say that the **coordinate vector** of **v** with respect to the basis set {$\mathbf{b'}_1$ … $\mathbf{b'}_7$, $\mathbf{w8}_0$, $\mathbf{w8}_1$ …w87} is:

3	9/2	8	4	0	5/2	9/2	0	-2	-1	2	-2	0	-1	-1	0

In the exact similar manner, we can find coordinate vectors foe every decimation till the image gets down-sampled into 1-pixel image. The calculation for that is shown in Image 3.

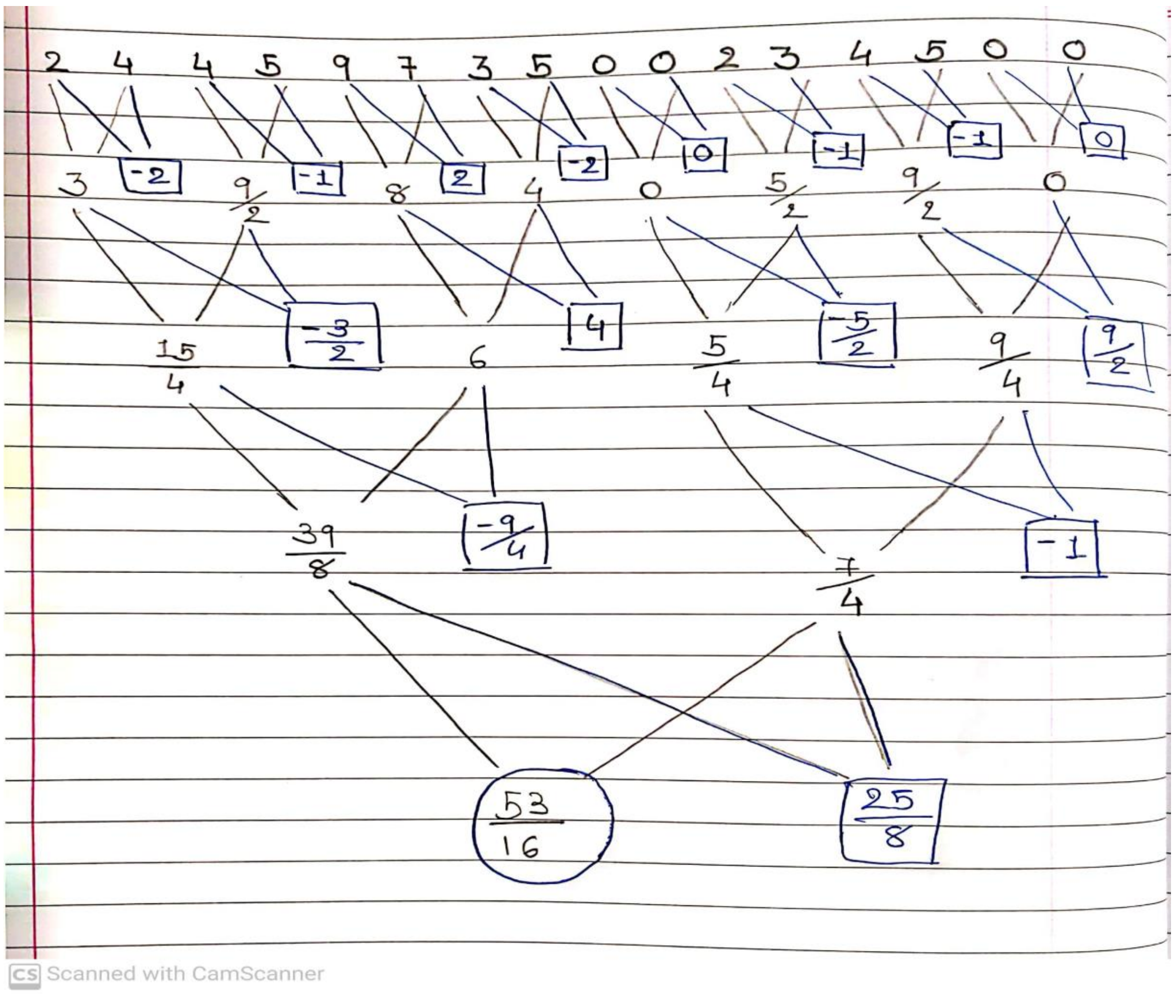


Image 3

Hence, 15 wavelet coefficients (marked with a square in Image 3) plus the overall average intensity i.e. intensity of one-pixel image form **wavelet transform** for $\mathbf{V}_{16}$. These 15 wavelet coefficients will be the coefficients for the wavelet vectors $\mathbf{w8}_0$, $\mathbf{w8}_1$ …w87, and $\mathbf{w4}_0$, $\mathbf{w4}_1$, $\mathbf{w4}_2$, **w43**, and $\mathbf{w2}_0$, $\mathbf{w2}_1$, and $\mathbf{w1}_0$.

Normalisation

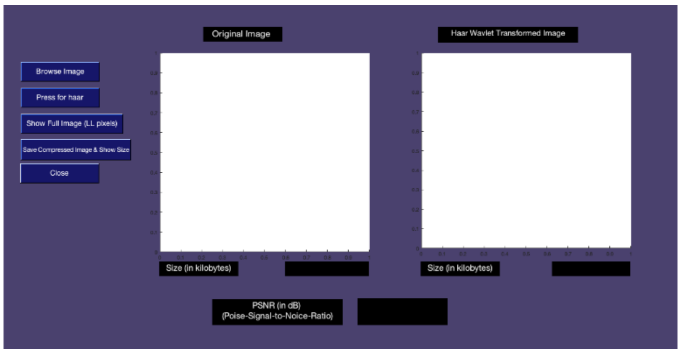
Since the “new” basis vectors we obtained via this process are orthogonal, and not **Orthonormal,** we can convert them into one by **multiplying and dividing each term in the linear combination representation by the norm of corresponding vector.** Therefore, the **new coordinates with respect to the Orthonormal basis will norm of the corresponding vectors time the old coordinates.**

So here, we have explained the Linear Algebra concepts for a base of one-dimensional image compression. The analogical method for two-dimensional image compression is described in the section “Coding and Simulation”. Since, we have not lost any information throughout this process of 1-D image processing, it is a lossless process. However, while practically compressing a 2-D image the redundant data would be discarded, and hence the compression would be a **lossy** one. (Details given in “Coding and Simulation section”)

3. Coding and Simulation:

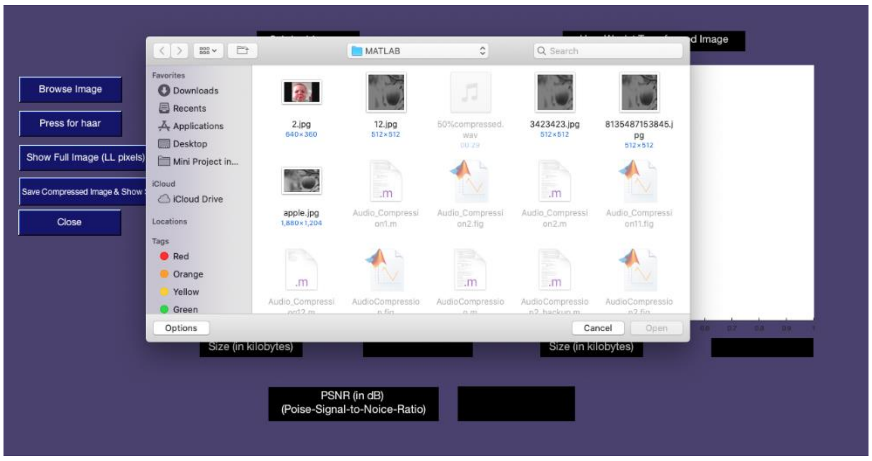
Coding strategy:

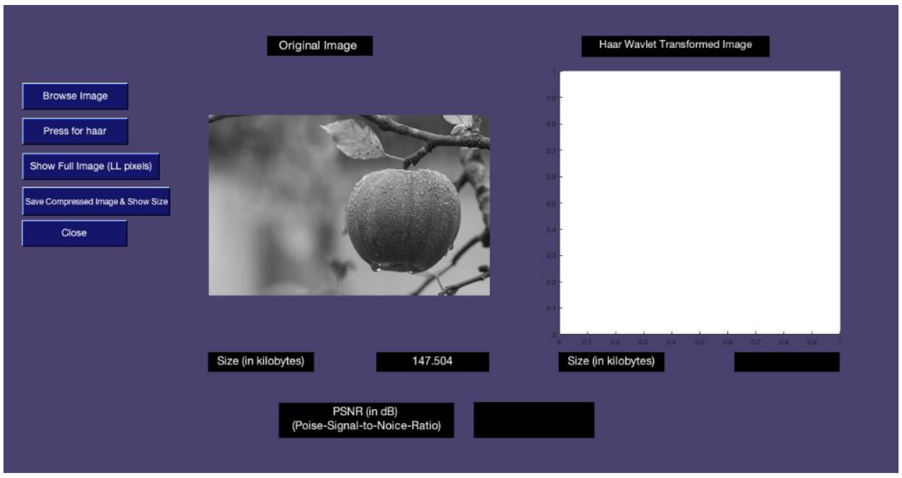
The Coding Strategy implemented involves using an Iterative type with count-controlled execution on a numerical computing programming environment MATLAB. The code also includes a Graphical User Interface (GUI), and runs as follows:-


Original Image
Haar Wavlet Transformed Image
Browse Image
Press for haar
Show Full Image (LL pixels)
Save Compressed Image & Show Size
Close
Size (in kilobytes)
Size (in kilobytes)
PSNR (in dB)
(Poise-Signal-to-Noice-Ratio)

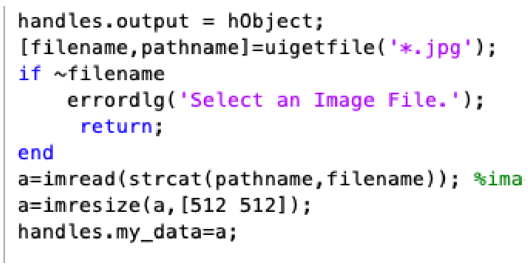
1. On executing the "main.m" file, an input in form of image (.jpg) is taken from the user specified location.
   After selecting the image, it is read by the MATLAB library function "imread".

   The image is plotted using the MATLAB library "imshow" function and "filename.bytes" returns the size of the image.




Original Image
Haar Wavlet Transformed Image
Browse Image
Press for haar
Show Full Image (LL pixels)
Save Compressed Image & Show Size
Close
Size (in kilobytes)
147.504
Size (in kilobytes)
PSNR (in dB)
(Poise-Signal-to-Noice-Ratio)

(i) "imread" MATLAB function [7]:

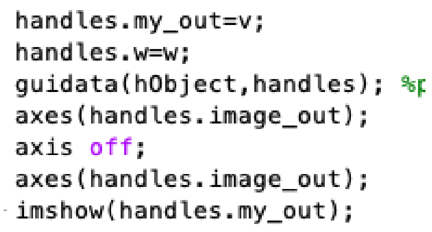


```
handles.output = hObject;
[filename,pathname]=uigetfile('*.jpg');
if ~filename
    errordlg('Select an Image File.');
     return;
end
a=imread(strcat(pathname,filename)); %ima
a=imresize(a,[512 512]);
handles.my_data=a;
```

- Reads Greyscale image as data from the file.
- Returns a two-dimensional array (mxn matrix in our case) array
- Usually of size 8-bit.

(ii) “imshow” MATLAB Function[8]:

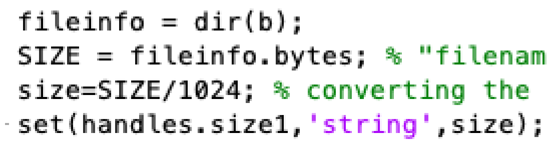
- After extracting the image data from a variable, this function plots the image.
- The image is plotted at 100% magnification by default.



```
handles.my_out=v;
handles.w=w;
guidata(hObject,handles); %p
axes(handles.image_out);
axis off;
axes(handles.image_out);
imshow(handles.my_out);
```

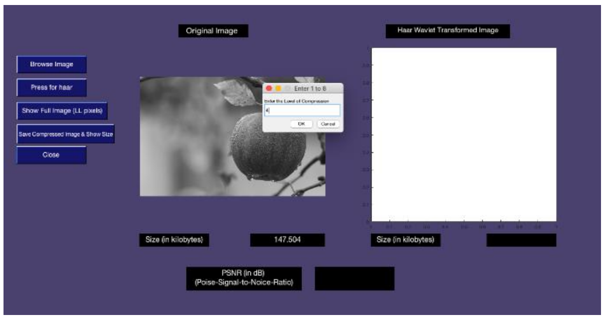
(iii) “filename.bytes” MATLAB Function (for original image):

- ‘.bytes’ extracts the image size in bytes.
- For fewer complications, we convert it to kilobytes (kb).
- The size of the original image is then projected to the I/O Screen.



```
fileinfo = dir(b);
SIZE = fileinfo.bytes; % "filenam
size=SIZE/1024; % converting the
set(handles.size1,'string',size);
```

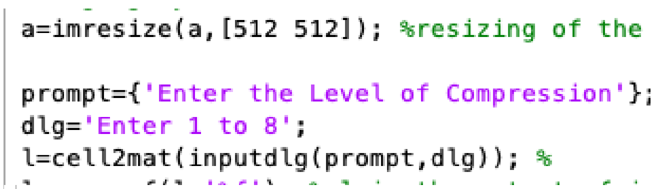
2. The option “Press for Haar” will implement the user-defined function “haar_DWT(a)”, where a is the variable holding the image data from “imread”.
   The user has to input the “level of Compression”.


Original Image
Haar Wavlet Transformed Image
Browse Image
Press for haar
Show Full Image (LL pixels)
Save Compressed Image & Show Size
Close
Enter 1 to 8
Enter the Level of Compression
4
OK
Cancel
Size (in kilobytes)
147.504
Size (in kilobytes)
PSNR (in dB)
(Poise-Signal-to-Noice-Ratio)

(iv) User-Defined “[v, w] =haar_dwt (a)”

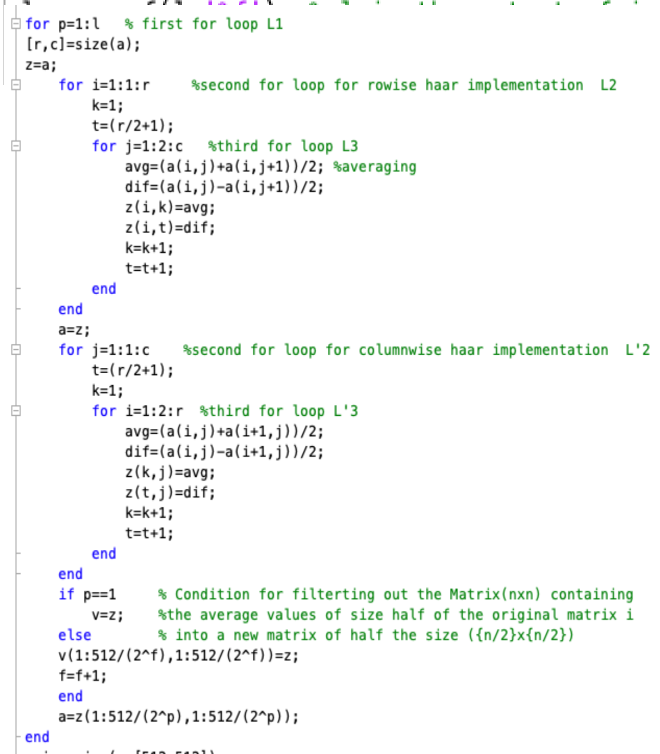
- Defined to perform the image compression by using the Haar Wavelet Approach.
- Using the “imresize” (MATLAB library function which makes the image data stored in ‘a’ in the form of 512x512 two-dimensional array).

- We will consider the 512x512 Two-Dimensional Arrays a matrix of 512x512 square-matrixes.
- The ‘level of compression’ is assigned to ‘l’, which will determine



```
a=imresize(a,[512 512]); %resizing of the

prompt={'Enter the Level of Compression'};
dlg='Enter 1 to 8';
l=cell2mat(inputdlg(prompt,dlg)); %
```



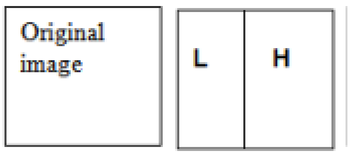
```
for p=1:l   % first for loop L1
[r,c]=size(a);
z=a;
    for i=1:1:r     %second for loop for rowise haar implementation  L2
        k=1;
        t=(r/2+1);
        for j=1:2:c   %third for loop L3
            avg=(a(i,j)+a(i,j+1))/2; %averaging
            dif=(a(i,j)-a(i,j+1))/2;
            z(i,k)=avg;
            z(i,t)=dif;
            k=k+1;
            t=t+1;
        end
    end
    a=z;
    for j=1:1:c    %second for loop for columnwise haar implementation  L'2
        t=(r/2+1);
        k=1;
        for i=1:2:r  %third for loop L'3
            avg=(a(i,j)+a(i+1,j))/2;
            dif=(a(i,j)-a(i+1,j))/2;
            z(k,j)=avg;
            z(t,j)=dif;
            k=k+1;
            t=t+1;
        end
    end
    if p==1     % Condition for filterting out the Matrix(nxn) containing
        v=z;    %the average values of size half of the original matrix i
    else        % into a new matrix of half the size ({n/2}x{n/2})
    v(1:512/(2^f),1:512/(2^f))=z;
    f=f+1;
    end
    a=z(1:512/(2^p),1:512/(2^p));
end
```

the iterations of the ‘first for loop’ or L1.

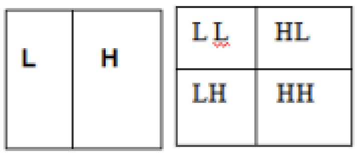
- The loops L2 and L3 handle the row-wise transformations & L’2 and L’3 handle the column-wise transformations. (working algorithm mentioned below)

Working of the function: (mathematical implementation of 2-D wavelet transform)

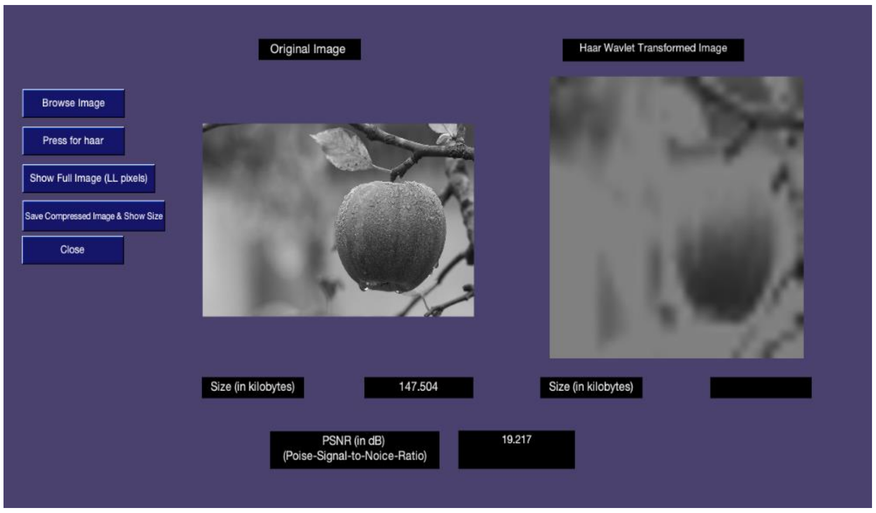
- As mentioned in the methodology, Haar Wavelet Transform involves majorly of two operations of averaging and differencing and the 2-dimensional array can be considered the orthogonal set of vectors inside the basis of the 512x512 vector space representing the image.
- Fetching each row element consecutively from the original matrix from “a” (or pair-wise as we did in one-dimensional array) and taking their average and storing their values in the dummy matrix “z” as portrayed in figure. (Average values of ‘a’ are stored in the part L of the ‘z’ matrix and similarly the difference in H). This is achieved by iterating L1 and L2 which transforms the Matrix ‘vertically’ into 2 halves.



- L and H are internally (or vertically) divided such that they occupy 512 x 256 parts of ’a’.
- Now the new matrix with average and differencing values is assigned to ‘a’ again.
- The loops L’1 and L’2 will fetch each column element in pairs and store their average (performing One-Dimensional Haar Transform) in LL and HL (horizontally) & difference values in LH and HH respectively as shown in figure. The L’1 and L’2 transforms the matrix ‘horizontally’



- As indicated in figure, the original matrix is divided into four 256 x 256 matrices in which LL is the one consisting of pair wise average values of each and every element from the original 512x512 matrix. The others (HL, LH, and HH) will only include the values which can be discarded or ignored.
- The “if” condition fetches the average values from ‘a’ matrix inside a new matrix of size half of 512x512.
- The whole transformation can take place ‘l’ times, (as controlled in loop L1) which ensures that the original matrix gets down-sampled by a factor of 2 after every iteration of ‘l’.


Original Image
Haar Wavlet Transformed Image
Browse Image
Press for haar
Show Full Image (LL pixels)
Save Compressed Image & Show Size
Close

Size (in kilobytes) 147.504

Size (in kilobytes)

PSNR (in dB)
(Poise-Signal-to-Noice-Ratio) 19.217

"Show Full Image (LL pixels)" plots the filtered 2-D array or matrix representing 2-Vector Space of compressed image.

Conceptual part of the function: (2-dimensional Haar Wavelet Transform)

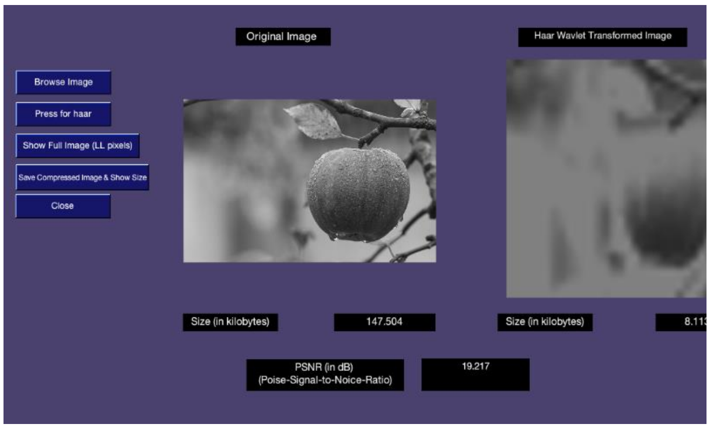
- As the 512 x 512 two-dimensional array is divided into 256 x 256 two-dimensional containing average values of the original matrix, it is the compressed or down-sampled two-dimensional which will be a 'set' of the original 2-D array and hence can represent the compressed image (as mentioned in One-Dimensional Array in "Methodology part").
- The 256 x 256 two-dimensional arrays now consists of smaller data and can further be down sampled into 128 x 128, 64 x 64 and so on till the Two-dimensional array becomes 1x1.
- **A noticeable factor here is that instead of achieving a lossless compression we get a Lossy compression.** This is due to the reason that the values in HL, LH and HH are either changed or discarded after every iteration and hence cannot be used in reconstruction of the original image by inverse transformation [9].
- Another interesting concept which can be observed here is that, any basis in 512 x 512 two-dimensional vector space can be obtained from any basis from each of 256 x 256, 128 x 128 and so on till 1x1 and taking their Union. This concept is applicable for one dimensional vector space using Haar Wavelet Transform and thus by analogy, can also be execute it for 2-D vector space by representing them as m x n matrix.
- Hence making this the most efficient greyscale image compression technique.[10]

Alternative implementation of 2-D Haar Wavelet approach using Matrix Multiplication (which doesn't work effectively for every image)

- It can be assumed that the image can be represented as n x n matrix (n in the power of 2).
- The matrix is then multiplied by an 8x8 Forward Transformation matrix [10], which splits the columns inside the matrix into two halves.

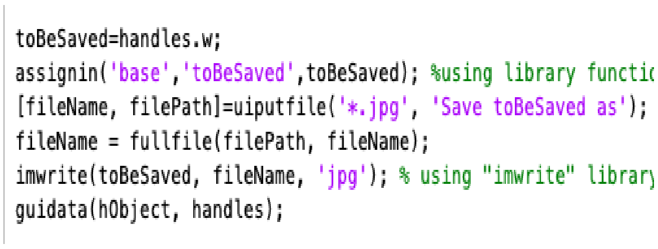
- Next step also involves the matrix to be multiplied by a 4x4 and 2x2 transform matrix consecutively. (same as we obtained after iterating for loops L1 and L2 and after using loop L’1 and L’2)
- The only drawback of this method is that it can only be performed on limited number of images as there exists limited Transformations Matrices of variable size.
- Hence, we didn’t use this method as our approach to solve the problem.

3. The compressed image can be saved using “imwrite” MATLAB Library Function. Accuracy metrics too can be saved using the user-defined “quality_measure.m” function as well as .bytes for Qualitative analysis between original and compressed image.


Original Image
Haar Wavlet Transformed Image
Browse Image
Press for haar
Show Full Image (LL pixels)
Save Compressed Image & Show Size
Close
Size (in kilobytes)
147.504
Size (in kilobytes)
8.11
PSNR (in dB)
(Poise-Signal-to-Noice-Ratio)
19.217

(v) “imwrite” MATLAB function:

- Takes image data as input variable
- Returns the image as a file in “.jpg” format.
- User will be able to specify the location and name of the compressed image.



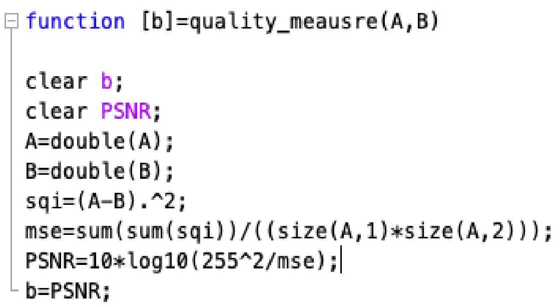
```
toBeSaved=handles.w;
assignin('base','toBeSaved',toBeSaved); %using library functi
[fileName, filePath]=uiputfile('*.jpg', 'Save toBeSaved as');
fileName = fullfile(filePath, fileName);
imwrite(toBeSaved, fileName, 'jpg'); % using "imwrite" librar
guidata(hObject, handles);
```

Qualitative Analysis factors:

(vi) User-Defined “quality_measure.m” functions:

- Reads the image data of original (m) and compressed (n) image in double data type. (m & n the respective matrices)
- First calculating the MSE (mean squared error) [1]

$$MSE = \frac{1}{mn}\sum_{y=1}^{m}\sum_{x=1}^{n}\left(I(x,y) - I^{1}(x,y)\right)^{2}$$



```
function [b]=quality_meausre(A,B)

clear b;
clear PSNR;
A=double(A);
B=double(B);
sqi=(A-B).^2;
mse=sum(sum(sqi))/((size(A,1)*size(A,2)));
PSNR=10*log10(255^2/mse);
b=PSNR;
```

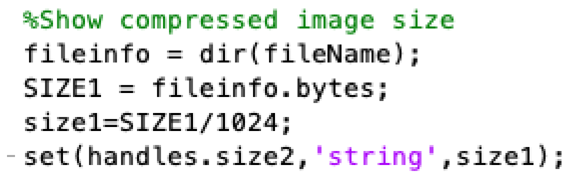
- And then the PSNR (poise-signal-to-noise ratio) [1]

$$PSNR(dB) = 20 \times \log_{10}\left(\frac{255}{\sqrt{MSE}}\right)$$

- If PSNR is between 25 to 35 it is considered an effective compression

(vii) "filename.bytes" Matlab Function (for compressed Image)

The size of the original image is known and now after fetching size of the compressed image, we can now compare the sizes of both images (in kb).



```
%Show compressed image size
fileinfo = dir(fileName);
SIZE1 = fileinfo.bytes;
size1=SIZE1/1024;
set(handles.size2,'string',size1);
```

Inferences: (This will also be implemented during the Presentation)

- For a user specified gray scale image, we performed 4 levels of compressions.
- The size of the input image was about 147 kb and the compressed image is 8 kb which is almost 16 times (or 2^4 times) compressed compared to the size of the original image.
- **This proves our thesis that at every Haar wavelet transformation, the matrix of the original image gets down-sampled by a factor of 2.**
- The PSNR obtained almost near to 20 shows that the image has maintained recognisable quality and also showcased high compressibility factor.
- **Another hypothesis which we stated in "approach" was that using Gray scale images would be better for this approach, as computation is faster and efficient. As opposed to using colour image (RGB) the computation which not only takes longer time but also the components get scattered during compression, which has a major reason behind opting Gray scale.**

4. Conclusion

The report was aimed at finding an algorithm for image compression (greyscale) for performing it very efficiently and effective. The approach we considered most suitable for such a problem definition was Haar Wavelet based image compression which utilizes the Linear Algebraic concepts like vector-space, change of basis, orthogonal set of vectors, etc. Using the concepts like 1-D and 2-D wavelet transform on the Haar basis and implementing them using iterative loops, matrix transformations, etc using MATLAB enables us to obtain a computationally efficient code which compresses a Greyscale image with very little running time and effort. The "Coding and Simulation" section showed that even the complicated concepts as explained above have many applications like image compression and can easily be implemented after breaking them down into simpler functionalities. Results obtained showed that this approach has a higher compressibility rate and still maintains quality standard (PSNR values above 15 dB) and hence leading to a Lossy compression.

We wanted to perform the comparative analysis between the Haar Wavelet Image compression and discrete cosine transform. However, due to time constraints we were unable to do so.

Closing remarks:

It was an absolute wholesome experience working through this project, spanning many different phases, from knowing nothing about Haar wavelet transform to knowing the non-complex nature of the approach we are not only dealing with but also learning how concepts from Linear Algebra can be applied or implemented with sheer perfection in implementing the solution for the 'Problem Definition'. Also realising the small factors such as taking into account some assumptions and at last proving them correct.

Individual contributions to the project of each member

Basically, all of us are aware of every domain covered in this report. However, the work division was in such a manner that Vishwa Raval (AU1940131) and Dev Jani (AU1940045) have contributed mostly in the part "Linear Algebra Concepts used" i.e. Mathematics behind 1-D and 2-D compression. Darsh Patel (AU19401501) and Yashvi Navadia (AU1940131) have mainly contributed in the "Coding and Simulation" section.

5. References

[1]. K. H. Talukder and Koichi Harada, Haar Wavelet Based Approach for Image Compression and Quality Assessment of Compressed Image: IAENG International Journal of Applied Mathematics, 1 February 2007.

[2] C. Mulcahy, Image Compression Using the Haar Wavelet Transform: Spelman Science and Math Journal

[3] Pixel Values

https://homepages.inf.ed.ac.uk/rbf/HIPR2/value.htm

[4] Greyscale Images

http://homepages.inf.ed.ac.uk/rbf/HIPR2/gryimage.htm

[5] Haar Wavelet Image Compression

https://people.math.osu.edu/husen.1/teaching/572/image_comp.pdf

[6] Coding the matrix: linear algebra through applications to computer science

Klein - Newtonian Press – 2015

[7] **http://matlab.izmiran.ru/help/techdoc/ref/imread.html**

[8] **https://www.mathworks.com/help/images/display-an-image-in-a-figure-window.html**

[9] P. Raviraj and M.Y. Sanavullah, The Modified 2D-Haar Wavelet Transformation in Image Compression: Middle-East Journal of Scientific Research 2 (2): 73-78, 2007 ISSN 1990-9233

https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.128.9190&rep=rep1&type=pdf

[10] A. Kulkarni and A. Junnarkar, Gray-Scale Image Compression Techniques: A Review: International Journal of Computer Applications (0975 – 8887) Volume 131 – No.13, December 2015
https://www.ijcaonline.org/research/volume131/number13/kulkarni-2015-ijca-907519.pdf#:~:text=After%20the%20study%20of%20gray,efficiently%20in%20producing%20compressed%20data.