# BIRLA INSTITUTE OF TECHNOLOGY & SCIENCE, PILANI

Work Integrated Learning Programmes Division


**M.Tech. in Data Science and Engineering**

# Change of Basis and Image Decomposition using Linear Algebra Techniques

DISSERTATION

Submitted in partial fulfillment of the requirements of the
MTech Data Science and Engineering Degree programme

By
Anish Arya
2020sc04930

Under the supervision of
Aayush Agarwal

# Table of Contents

BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE, PILANI

CERTIFICATE

This is to certify that the Dissertation entitled **Change of Basis and Image Decomposition using Linear Algebra Techniques** and submitted by **Mr. Anish Arya**, BITS ID. No. **2020SC04930** in partial fulfillment of the requirements of Dissertation (**DSECLZG628T**), embodies the work done by him under my supervision.

*Signature of the Supervisor*
Place: Gurugram
Date: 12 - March - 2023

Aayush Agarwal
Senior Consultant
Fractal Analytics

*“There's always an algorithm outside your comfort zone with lesser runtime complexity and greater scalability to solve a particular problem.”*

–Anish Arya

**BIRLA INSTITUTE OF TECHNOLOGY SCIENCE,**

**PILANI**

**DSECLZG628T DISSERTATION**

| | |
|---|---|
| Dissertation Title | : Change of Basis & Image Decomposition using Linear Algebra |
| Name of Supervisor | : Aayush Agarwal |
| Name of Student | : Anish Arya |
| ID of the Student | : 2020sc04930 |

## Abstract

Video cameras capture data in a poor format for transmitting video. To effectively transmit video with low latency and lossy compression so that the machine (generally speaking, an ADAS or advanced driver-assistance system) can make an accurate decision without sacrificing any precision, the basis is altered using linear algebra. What "basis" is best for video and picture compression to handle these kinds of events, however, is a crucial question that hasn't received a thorough answer.

Albeit Elon Musk's Tesla, Inc. is a well-known automobile manufacturer and is making excellent strides in the development of ADAS vehicles, there is still a need for research employing linear algebra to address the issue of making accurate and precise decisions quickly while using less computing power. The amount of space that can be used to store sizable amounts of data (in the form of files) on the servers of large organizations or on the hard drives of computer systems is diminishing faster than the rate at which data must be stored. As a result, there is a high need for compression techniques that can assist in reducing the size of data files. In this project, we'll demonstrate how linear algebra may be used to compress images in a time and space-effective manner. We'll basically discuss how the SVD (Singular Value Decomposition) and Haar Wavelet algorithms are frequently used to compress images, hence reducing computer memory use and response time.

A report by renowned consulting firm McKinsey & Co. claims that the widespread use of self-driving cars may prevent up to $190 billion in annual damages and medical expenses and 90% of all traffic accidents in the United States.

***Source:*** *https://www.wsj.com/articles/will-we-blame-self-driving-cars-11674745636*

**Keywords:**

Singular Value Decomposition, Haar Wavelet Transformation, Progressive Transmission, Rayleigh Quotient, Image Spectrum, Normalization, Approximation of a Matrix, Lossy Compression

# Acknowledgements

I want to start by thanking Mr. Aayush Agarwal, my M.Tech supervisor, for his insightful counsel and continuous support during my studies. I also like to thank Ms. Anitha N for her assistance and technical advice, without which finishing this work could have been an impossible task. I want to express my gratitude to all of my team members for their unending support and encouragement. They provided me with incredible support, without which I could not have finished my studies. Finally, I want to express my gratitude to my parents and friends for their unwavering emotional support during my academic career.

# List of Figures

# Abbreviations

| | |
|---|---|
| **SVD** | **S**ingular Value Decomposition |
| **HWTM** | Haar Wavelet Transformation of a matrix |

# Chapter 1

## Introduction

### 1.1 Solution Description

The following ideas would be included in this solution:

a. SVD-based image compression (Singular Value Decomposition)
b. Haar Wavelet Transformation
c. Progressive Transmission
d. Normalization Technique

### 1.2 Business Impact

The fundamental idea is that each image can be represented as a matrix, which can then be transformed using linear algebra (SVD in the mid-semester report and Haar Wavelet Transformation in the final report) to produce a reduced and normalized matrix. **This corresponds to an image that requires significantly less storage space than the original image and such kind of basis change without compromising the accuracy and precision of the decision made by an ADAS system.**

### 1.3 Fundamental Design

The amount of space available to store significant amounts of data (in the form of files) on the hard drives of computer systems or onto the servers of large corporations is decreasing at a faster rate than the amount of data that needs to be stored. As a result, many compression techniques that can help to reduce the size of data files are in demand. In this project, we'll talk about how linear algebra can be applied to picture compression. In essence, we'll talk about how SVD and Wavelet algorithms are widely used to compress images, saving computer memory in the process.The core concept is that each image can be represented as a matrix, and that matrix may be reduced by applying linear algebra (SVD and Wavelet) on it. The image that corresponds to this reduced matrix uses significantly less storage space than the original image did.

# Chapter 2

## Image Decomposition using Singular Value Decomposition(SVD)

A m x n matrix format can be used to represent an image with m x n pixels. Assume we have a 3000 x 3000 pixel (or 3000 x 3000 matrix) grayscale image that is 9 megapixels in size. The amount of black and white for each pixel is determined by a number between 0 and 255, with 0 being black and 255 denoting white. An 8.6 Mb image is produced by storing approximately 1 byte for each of these integers (and consequently each pixel). A color image typically consists of three colours: red, green, and blue (RGB). Each of them is represented by a matrix, therefore colour images take up three times as much space to store (25.8 Mb).

### 2.1 Rayleigh Quotient Formulation and second-norm of a Matrix

The Rayleigh quotient R(M,x), for a given complex Hermitian matrix M and nonzero vector x, is defined as:

$$R(M,x) = x^{*}Mx/x^{*}x$$

For real matrices and vectors, the symmetry requirement replaces the Hermitian condition, and the conjugate transpose $x^{*}$ becomes the standard transposition $x^{T}$. Keep in mind that for any real scalar c!= 0, R(M, cx) = R(M, x). Remember that real eigenvalues exist in a Hermitian (or real symmetric) matrix. It can be demonstrated that when x is $v_{min}$(the corresponding eigenvector)., the Rayleigh quotient for a given matrix reaches its minimal value, $\lambda_{min}$ (the least eigenvalue of M). $R(M, x) \leq \lambda_{max}$ and $R(M, v_{max}) = \lambda_{max}$ behave similarly. To get the precise values of all eigenvalues, the Rayleigh quotient is employed in the min-max theorem. It is also used in eigenvalue algorithms to convert an approximate eigenvalue to an approximate eigenvector. This serves as the basis for Rayleigh quotient iteration, specifically.

1. A vector's 2-Norm is provided by:

$$\| x \| = \mathrm{sqrt}(\Sigma^{n}_{i=1} x_i^2)$$

2. Matrix 2-norm is the subordinate to the vector 2-norm:
    a. $\| A \|_2$ is the A∗A's maximum eigenvalue
    b. The matrix 2-norm is referred to as the spectral norm because of its relationship to eigenvalues.

c. Note that A∗A is n x n and Hermitian for any given m x n matrix. Real values make up A's eigenvalues. Moreover, A is at least semi-definitely positive as X*A*A*X = (AX)*AX ≥ 0∀X. As a result, the eigenvalues of A*A are real-valued and non-negative; denote them as

$$\sigma 1 \text{ squared} \geq \sigma 2 \text{ squared} \geq \sigma 3 \text{ squared} \geq \ldots \geq \sigma n \text{ squared} \geq 0$$

d. Keep in mind that the size of these eigenvalues is listed, with σ1 squared being the greatest. The singular values of matrix A are these eigenvalues. These eigenvalues are matched by n orthonormal (therefore independent) eigenvectors U1, U2,..., Un with

$$(A^*A)U_k = (\sigma_k^2)U_k, \ \forall \ \ 1 \leq k \leq n$$

e. The columns of a unitary n x n matrix U are n eigenvectors, which diagonalizes matrix A*A under similarity (diagonal matrix $U^*(A^*A)U$) has eigenvalues on the diagonal.

f. The n eigenvectors U1, U2,..., Un can be used as a basis because they are independent, and vector X can be written as

$$X = \Sigma^n_{k=1} c_k U_k$$

where the $c_k$ are constants that rely on X-dependent constants. To get the below equation, multiply A*A by X:

$$A^*AX = A^*A(\Sigma^n_{k=1} c_k U_k) = \Sigma^n_{k=1} c_k \sigma_k^2 U_k$$

which get us to

$$\| AX \|^2 = (AX)^*AX = X^*(A^*AX) = (\Sigma^n_{k=1} c_k^* U_k^*)(\Sigma^n_{j=1} c_j \sigma_j^2 U_j = \Sigma^n_{k=1} |c_k|^2 \sigma_k^2 \leq \sigma_1^2(\Sigma^n_{k=1} |c_k|^2)$$
$$= \sigma_1^2 \| X \|^2$$

for any X. As a result, we have finished step 1.: We discovered a fixed $K = \sigma_1^2$ such that $\| AX \|_2 \leq K \| X \|_2$

We must identify at least one vector $X_0$ for which equality holds in step 2; i.e. Finding an X with the characteristic that $\| AX \| = K \| X \|$ It will operate with the unit-length eigenvector linked to eigenvalue $\sigma_1^2$. Hence, the matrix 2-norm is given by $\| A \|_2 = \text{sqrt}(\sigma_1^2)$, the sqrt of the maximum eigenvalue of $A^*A$.

## 2.2 SVD

Given Assume $A \in C^{m\times n}$.(A may not always be of full rank, and m and n may be arbitrary), a SVD of A is a factorization $A=U\Sigma V^*$ where $\Sigma \in C^{m\times n}$ is diagonal, $U \in C^{m\times m}$ and $V \in C^{n\times n}$ are unitary.

The diagonal entries $\sigma_j$ of $\Sigma$ are also assumed to be nonnegative and in a non-increasing order, i.e., $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_p \geq 0$, where p=min (m,n) U and V are always square unitary matrices, but the diagonal matrix has the same shape as A even if A is not square.

## 2.3 Approximation of a Matrix using SVD

By a linear combination of the singular values of the matrix A and the related vectors $u_i$ and $v_i^T$, a m n matrix A can be expressed ($v_i^T$ is $i^{th}$ column of V and where $u_i$ is $i^{th}$ column of U)

a. $A=U\Sigma V^T$

b. Rewrite A as following:

   i. $A = u_1\sigma_1 v_1^T + u_2\sigma_2 v_2^T + \ldots + u_i\sigma_i v_i^T + \ldots + u_n\sigma_n v_n^T$
   ii. $\Rightarrow A = \sigma_1 u_1 v_1 + \sigma_2 u_2 v_2 + \ldots + \sigma_i u_i v_i + \ldots + \sigma_n u_n v_n$

c. The $\{\sigma_1 u_1 v_1, \sigma_2 u_2 v_2 2, \ldots, \sigma_n u_n v_n\}$ terms are listed in decreasing order of dominance (as $\sigma_1 \geq \sigma_2 \geq \cdots\cdots \geq \sigma_n$).

d. By lowering the number of iterations in the linear combination, one can approximate the matrix A:

   i. $A_i = \sigma_1 u_1 v_1 + \sigma_2 u_2 v_2 + \ldots + \sigma_i u_i v_i$

e. This symbolize:

   i. $A_i = \sigma_1 u_1 v_1 + \sigma_2 u_2 v_2 + \ldots + \sigma_i u_i v_i$

g. Lower image quality but less storage requirements could be obtained by keeping only part of these terms. Principal Component Analysis is another name for this method (PCA).

## 2.4 Saving Memory by Decomposition using SVD

a. There are $n^2$ elements needed for matrix A. $\Sigma$ requires n elements and both $V^T$ and U require $n^2$ elements each. Hence, $2n^2 + n$ items are needed to store the entire SVD. $(2n + 1)$ elements are needed to maintain one term in the SVD, $\sigma_1 u_1 v_1^T$.

b. If the first $i < n/2$ terms are kept, then storing the reduced matrix only needs $i(2n + 1)$ elements, which is way lesser than the total number of elements in the matrix $(2n^2 + n)$ as n grows.

c. We obtain the reduced matrix as a result of PCA; in other words, the members of this reduced matrix include the appropriate pixels from the compressed image.

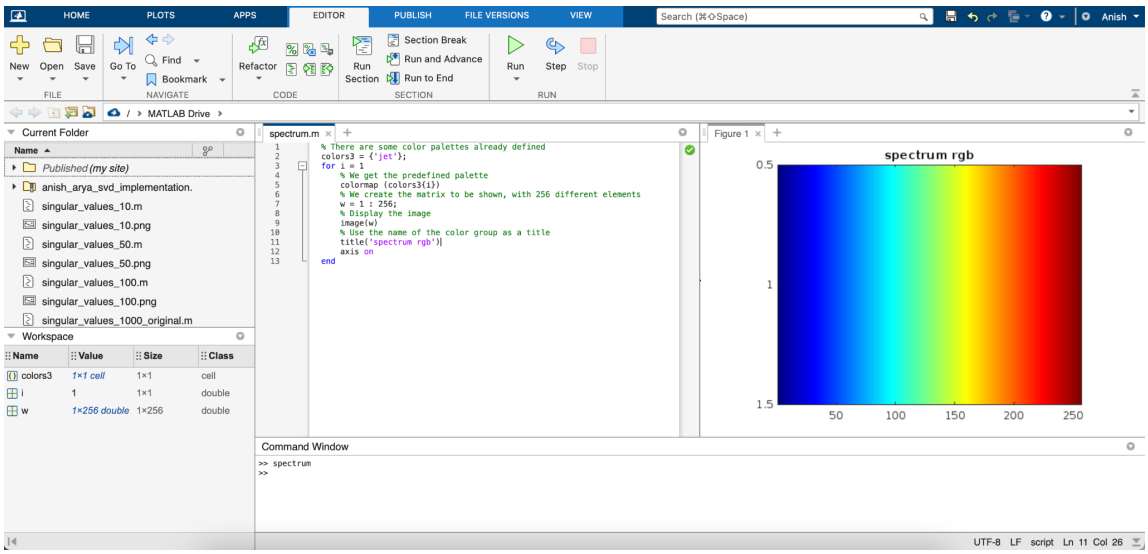
## 2.5 Spectrum of an Image using Matlab

a. A quick explanation of how Matlab creates images is required in order to comprehend how SVD is utilized in image compression.

b. Every item in the matrix basically corresponds to a tiny square of the image. In Matlab, the entry's numerical value corresponds to a color.

c. By entering the following code in Matlab, the color spectrum can be viewed:

**Figure 2.1: Spectrum of an Image using Matlab**



d. In light of this, a picture of nine square blocks that make up one large block should be produced by entering a 3 × 3 matrix of random integers. Moreover, each individual block's color will match the color depicted at that numerical value.

e. It has been demonstrated that while computing the linear combinations defining S, any matrix S can be approximated with a smaller number of iterations. The Matlab command "svd" can be used to display this. Matlab displays the original image created by S and the image created by an approximation of S.

## 2.6 SVD Example using an Image

a. On the few pages that follow before Haar-Wavelet transformation, we take an image and explore the effects that several iterations will have on it:

**Figure 2.2: 1000 Singular Values == Grayscale of Original Image**

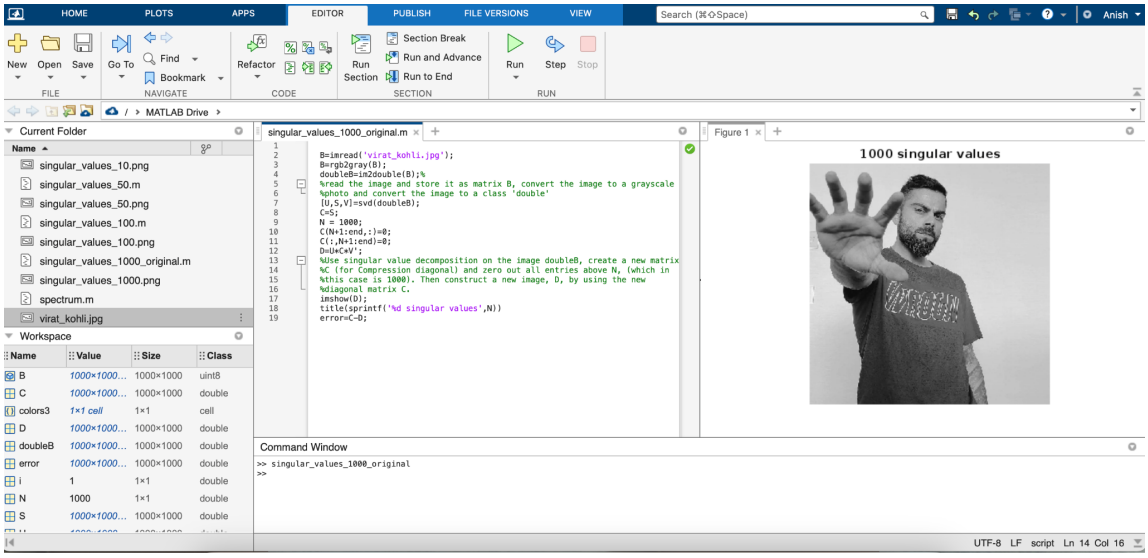


**Figure 2.3: 100 Singular Values**

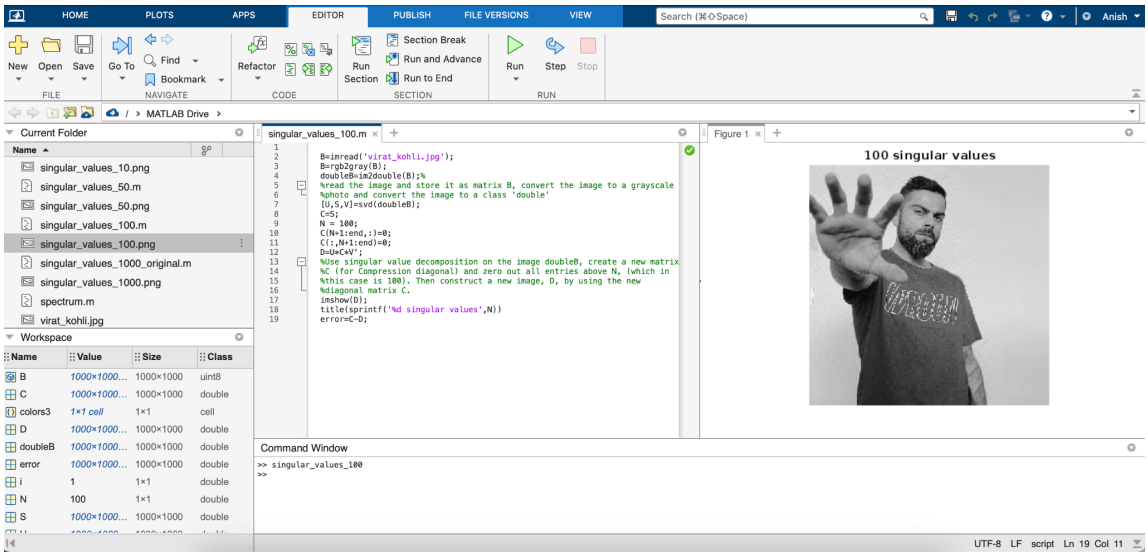



**Figure 2.4: 50 Singular Values**

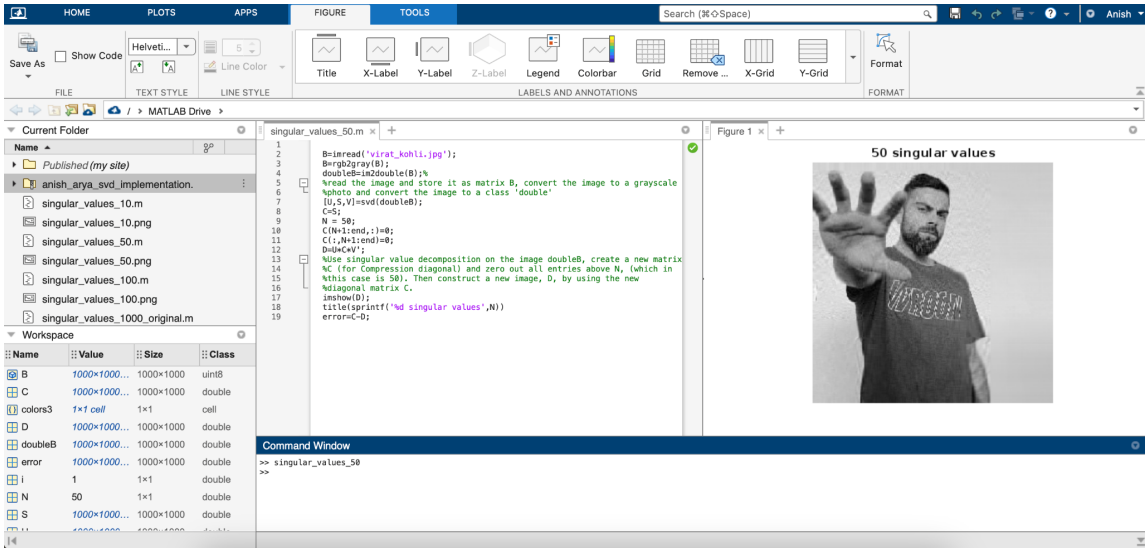
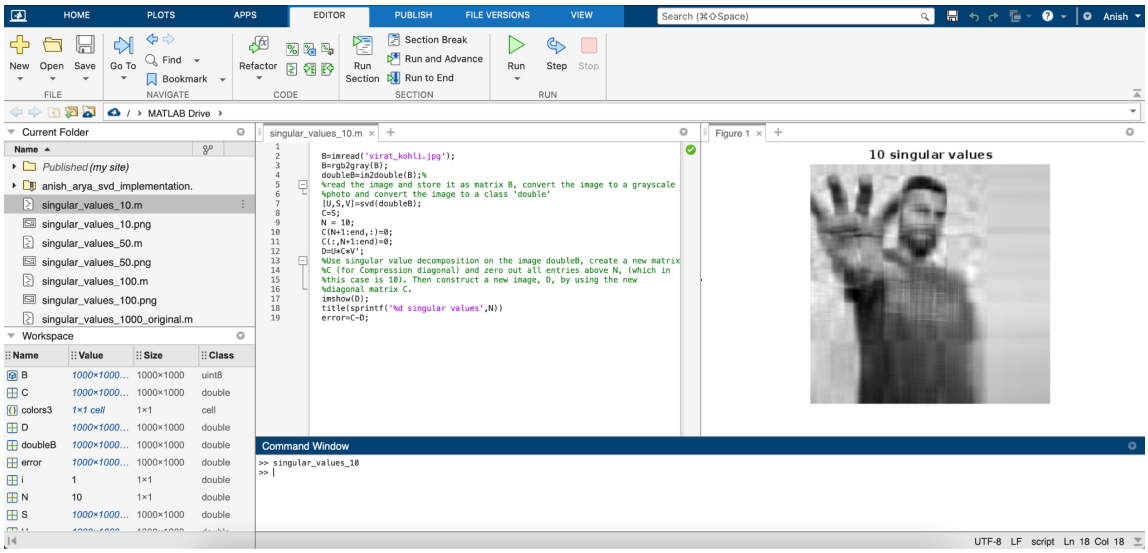


**Figure 2.5: 10 Singular Values**



# Chapter 3

## Haar Wavelet Transformation of a Matrix(HWTM)

### 3.1 Introduction

**Definition:** A straightforward technique of compression known as the Haar Wavelet Transformation involves averaging and diffencing terms, saving detail coefficients, removing data, and reconstructing the matrix so that it resembles the original matrix.

Here, we'll talk about wavelet compression, another method for compressing images.

An effective method for both lossless and lossy image compression is the Haar wavelet algorithm. It creates a sparse or nearly sparse matrix by averaging and diffencing data in an image matrix. A matrix is said to be sparse if a sizable number of its entries are 0. Smaller file sizes can be achieved by effectively storing sparse matrices.

A data encoding technique known as "lossy" compression in information technology compresses data by throwing away (losing) some of it. The process seeks to reduce the quantity of data that a computer must hold, handle, and/or communicate.

A type of data compression method known as "lossless data compression" enables the exact reconstruction of the original data from the compressed data. Contrarily, lossy data compression only allows for the approximate reconstruction of the original data, even though it typically results in higher compression rates (and therefore smaller sized files).

While rgb (color) photos can be handled by compressing each of the color layers separately, we focused on grayscale photographs in our effort. Beginning with a picture A, which may be thought of as an m-n matrix with values ranging from 0 (for black) to 255 (for white). This would be an unsigned 8-bit integer matrix in Matlab. This image is then divided into 8x8 blocks with any necessary padding. We work with these 8 by 8 blocks.

## 3.2 Example

An image of the flying buttresses of Paris's Notre Dame Cathedral in grayscale, measuring 512 by 512 pixels, is shown below:

**Figure 3.1: Grayscale Image of Flying buttresses of Notre Dame de Paris**

The upper left 8 x 8 area of the image is as follows:

$$A = \begin{bmatrix} 88 & 88 & 89 & 90 & 92 & 94 & 96 & 97 \\ 90 & 90 & 91 & 92 & 93 & 95 & 97 & 97 \\ 92 & 92 & 93 & 94 & 95 & 96 & 97 & 97 \\ 93 & 93 & 94 & 95 & 96 & 96 & 96 & 96 \\ 92 & 93 & 95 & 96 & 96 & 96 & 96 & 95 \\ 92 & 94 & 96 & 98 & 99 & 99 & 98 & 97 \\ 94 & 96 & 99 & 101 & 103 & 103 & 102 & 101 \\ 95 & 97 & 101 & 104 & 106 & 106 & 105 & 105 \end{bmatrix}$$

We'll pay special attention to the first row:

$$r_1 = [\ 88\ 88\ 89\ 90\ 92\ 94\ 96\ 97]$$

We'll undergo the change in five steps:

**Step 1:** The first step is to link up every column as follows:

$$[88, 88], [89, 90], [92, 94], [96, 97]$$

**Step 2:** The average of these pairs should be used as a replacement for the first four columns of $r_1$, and the difference between these pairs should be used for the final four columns of $r_1$. This new row will be known as $r_1h_1$.

$$r_1h_1 = [\ 88\ 89.5\ 93\ 96.5\ 0\ -0.5\ -1\ -0.5]$$

The **approximation coefficients** are the first four entries, and the **detail coefficients** are the final four.

**Step 3:** The first four columns of this new row are then grouped as below:

$$[88, 89.5], [93, 96.5]$$

**Step 4:** Now swap out the first two $r_1h_1$ columns for the average of the pairs, followed by the following two columns for half of the difference between the pairs. The final four rows of $r_1h_1$ remain unaltered. This second new row will be designated as $r_1h_1h_2$:

$$r_1h_1h_2 = [88.75\ 94.75\ -0.75\ -1.75\ 0\ -0.5\ -1\ -0.5]$$

**Step 5:** The final step is to combine the first two entries of $r_1h_1h_2$:

$$[88.75, 94.75]$$

Now, The average of the pairings should be used in place of the first column of r1h1h2 and half of the difference between the pairs should be used in place of the second column. The final six rows of $r_1h_1h_2$ remain unchanged. This final new row will be designated as $r_1h_1h_2h_3$:

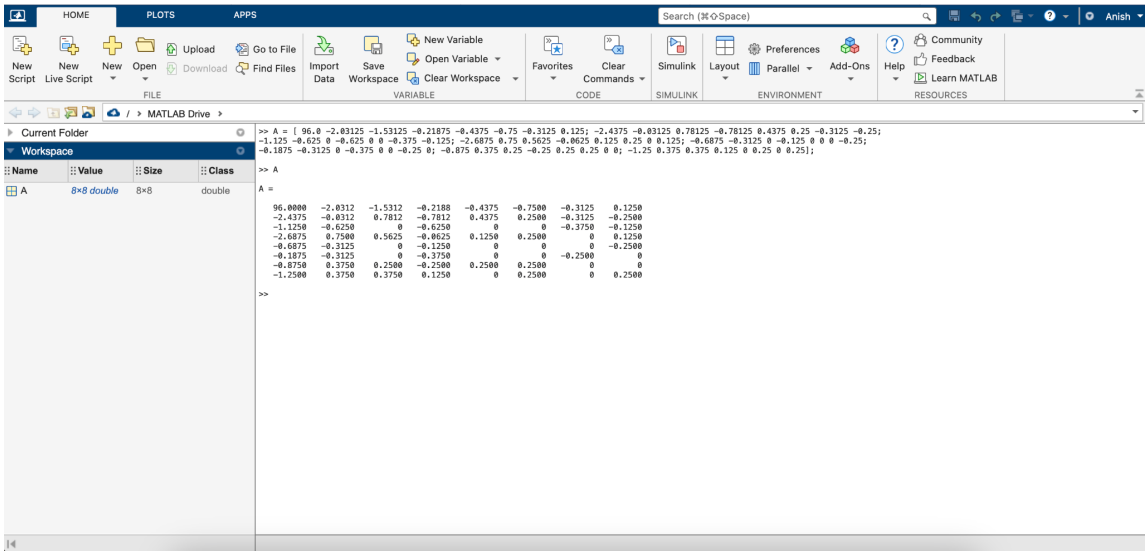
$r_1h_1h_2h_3 = [\ 91.75\ -3\ -0.75\ 2\ -1.75\ 0\ -0.5\ -1\ -0.5]$

The following rows of row A are then subjected to the same procedure. Then, grouping rows similarly to grouping columns, we repeat this operation for the columns of A. The matrix as a result is:

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| **96.0000** | **-2.0312** | **-1.5312** | **-0.2188** | **-0.4375** | **-0.7500** | **-0.3125** | **0.1250** |
| **-2.4375** | **-0.0312** | **0.7812** | **-0.7812** | **0.4375** | **0.2500** | **-0.3125** | **-0.2500** |
| **-1.1250** | **-0.6250** | **0** | **-0.6250** | **0** | **0** | **-0.3750** | **-0.1250** |
| **-2.6875** | **0.7500** | **0.5625** | **-0.0625** | **0.1250** | **0.2500** | **0** | **0.1250** |
| **-0.6875** | **-0.3125** | **0** | **-0.1250** | **0** | **0** | **0** | **-0.2500** |
| **-0.1875** | **-0.3125** | **0** | **-0.3750** | **0** | **0** | **-0.2500** | **0** |
| **-0.8750** | **0.3750** | **0.2500** | **-0.2500** | **0.2500** | **0.2500** | **0** | **0** |
| **-1.2500** | **0.3750** | **0.3750** | **0.1250** | **0** | **0.2500** | **0** | **0.2500** |

There are several 0 entries in this resulting matrix, and the majority of the other entries are near to 0. **Due to differencing and the fact that adjacent pixels in an image typically do not differ significantly from one another, this is the outcome**. Following this, we will go over how to use matrix multiplication to this procedure.

**Figure 3.2: Matrix in Matlab (Just For Reference)**



## 3.3 HWTM

If we assume $\mathbf{H_1}$ =

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 0.5000 | 0 | 0 | 0 | 0.5000 | 0 | 0 | 0 |
| 0.5000 | 0 | 0 | 0 | -0.5000 | 0 | 0 | 0 |
| 0 | 0.5000 | 0 | 0 | 0 | 0.5000 | 0 | 0 |
| 0 | 0.5000 | 0 | 0 | 0 | -0.5000 | 0 | 0 |
| 0 | 0 | 0.5000 | 0 | 0 | 0 | 0.5000 | 0 |
| 0 | 0 | 0.5000 | 0 | 0 | 0 | -0.5000 | 0 |
| 0 | 0 | 0 | 0.5000 | 0 | 0 | 0 | 0.5000 |
| 0 | 0 | 0 | 0.5000 | 0 | 0 | 0 | -0.5000 |

If so, $AH_1$ is the same as applying the first step above to every row of A. Particularly,

$AH_1$ =

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 88.0000 | 89.5000 | 93.0000 | 96.5000 | 0 | -0.5000 | -1.0000 | -0.5000 |
| 90.0000 | 91.5000 | 94.0000 | 97.0000 | 0 | -0.5000 | -1.0000 | 0 |
| 92.0000 | 93.5000 | 95.5000 | 97.0000 | 0 | -0.5000 | -0.5000 | 0 |
| 93.0000 | 94.5000 | 96.0000 | 96.0000 | 0 | -0.5000 | 0 | 0 |
| 92.5000 | 95.5000 | 96.0000 | 95.5000 | -0.5000 | -0.5000 | 0 | 0.5000 |
| 93.0000 | 97.0000 | 99.0000 | 97.5000 | -1.0000 | -1.0000 | 0 | 0.5000 |
| 95.0000 | 100.0000 | 103.0000 | 101.5000 | -1.0000 | -1.0000 | 0 | 0.5000 |
| 96.0000 | 102.5000 | 106.0000 | 105.0000 | -1.0000 | -1.5000 | 0 | 0 |

Likewise, by defining H2 as follows:

$H_2$ =

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 0.5000 | 0 | 0.5000 | 0 | 0 | 0 | 0 | 0 |
| 0.5000 | 0 | -0.5000 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0.5000 | 0 | 0.5000 | 0 | 0 | 0 | 0 |
| 0 | 0.5000 | 0 | -0.5000 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 1.0000 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 1.0000 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 1.0000 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1.0000 |

If all of the rows of A are considered, then $AH_1H_2$ is equivalent to the two previous phases.

$AH_1H_2$ =

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 88.7500 | 94.7500 | -0.7500 | -1.7500 | 0 | -0.5000 | -1.0000 | -0.5000 |
| 90.7500 | 95.5000 | -0.7500 | -1.5000 | 0 | -0.5000 | -1.0000 | 0 |
| 92.7500 | 96.2500 | -0.7500 | -0.7500 | 0 | -0.5000 | -0.5000 | 0 |
| 93.7500 | 96.0000 | -0.7500 | 0 | 0 | -0.5000 | 0 | 0 |
| 94.0000 | 95.7500 | -1.5000 | 0.2500 | -0.5000 | -0.5000 | 0 | 0.5000 |
| 95.0000 | 98.2500 | -2.0000 | 0.7500 | -1.0000 | -1.0000 | 0 | 0.5000 |
| 97.5000 | 102.2500 | -2.5000 | 0.7500 | -1.0000 | -1.0000 | 0 | 0.5000 |
| 99.2500 | 105.5000 | -3.2500 | 0.5000 | -1.0000 | -1.5000 | 0 | 0 |

Lastly, if we define $H_3$:

$H_3$ =

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 0.5000 | 0.5000 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0.5000 | -0.5000 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1.0000 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1.0000 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 1.0000 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 1.0000 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 1.0000 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1.0000 |

Then $AH_1H_2H_3$ is identical to performing all three of the aforementioned processes on every row of A. More specifically:

$AH_1H_2H_3$ =

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 91.7500 | -3.0000 | -0.7500 | -1.7500 | 0 | -0.5000 | -1.0000 | -0.5000 |
| 93.1250 | -2.3750 | -0.7500 | -1.5000 | 0 | -0.5000 | -1.0000 | 0 |
| 94.5000 | -1.7500 | -0.7500 | -0.7500 | 0 | -0.5000 | -0.5000 | 0 |
| 94.8750 | -1.1250 | -0.7500 | 0 | 0 | -0.5000 | 0 | 0 |
| 94.8750 | -0.8750 | -1.5000 | 0.2500 | -0.5000 | -0.5000 | 0 | 0.5000 |
| 96.6250 | -1.6250 | -2.0000 | 0.7500 | -1.0000 | -1.0000 | 0 | 0.5000 |
| 99.8750 | -2.3750 | -2.5000 | 0.7500 | -1.0000 | -1.0000 | 0 | 0.5000 |
| 102.3750 | -3.1250 | -3.2500 | 0.5000 | -1.0000 | -1.5000 | 0 | 0 |

Suppose H is the result of the products of these three matrices ($H_1H_2H_3$):

H =

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| **0.2500** | **0.2500** | **0.5000** | **0** | **0** | **0** | **0** | **0** |
| **0.2500** | **0.2500** | **-0.5000** | **0** | **0** | **0** | **0** | **0** |
| **0.2500** | **-0.2500** | **0** | **0.5000** | **0** | **0** | **0** | **0** |
| **0.2500** | **-0.2500** | **0** | **-0.5000** | **0** | **0** | **0** | **0** |
| **0** | **0** | **0** | **0** | **1.0000** | **0** | **0** | **0** |
| **0** | **0** | **0** | **0** | **0** | **1.0000** | **0** | **0** |
| **0** | **0** | **0** | **0** | **0** | **0** | **1.0000** | **0** |
| **0** | **0** | **0** | **0** | **0** | **0** | **0** | **1.0000** |

* Please keep the following things in mind:

1. The orthogonal subset (vector space of 8 dimensions over IR (IR8)) is formed from the columns of $H_1$.
2. They serve as the **basis** of IR as a result. $H_1$ can be inverted as a result. With $H_2$ and $H_3$, the same holds true.
3. As H is a product of invertible matrices, its columns serve as the orthogonal basis for IR.
4. We only need to multiply A on the left by $H^T$ to apply the technique to the columns. As a result, the matrix is $H^TAH$.

The fact that this procedure is reversible is crucial. The invertible matrix H in particular if:

$$B = H^TAH$$

Then:

$$A = (H^T)^{-1}BH^{-1}-1$$

B is an illustration of A's compressed image (in the sense that it is sparse). Furthermore, this compression is lossless since H is invertible.

## 3.4 Lossy Compression

To conduct lossy compression and maintain the quality of the compressed image, use the Haar wavelet transform. Initially, the ratio of non-zero elements in the original to non-zero elements in the compressed image is known as the compression ratio.

Let A represent one of our 8 by 8 blocks and $H^TAH$ represent A's compressed Haar wavelet picture. Several detail coefficients in $H^TAH$ are almost zero. We'll choose a number (ε) greater than 0 and set all of the $H^TAH$ elements with absolute values greater than 0 to 0. This matrix is now more compressed and contains more 0 values. This image can be decompressed using the inverse Haar wavelet transform to produce a result that is nearly identical to the original.

**Compression ratio**

As the image is decompressed, some detail will be lost if we set our threshold value to be positive (i.e., greater than zero), as some entries of the converted matrix will be reset to zero. The important thing is to make a sensible choice so that the compression is carried out efficiently with the least amount of image damage. The ratio of nonzero entries in the transformed matrix ($B = H^TAH$) to the number of nonzero entries in the compressed matrix derived from B by applying the threshold ε is known as the compression ratio.

**Images for example:**

**Figure 3.3: Original Image (left) | 10:1 Compression Ratio (right)**

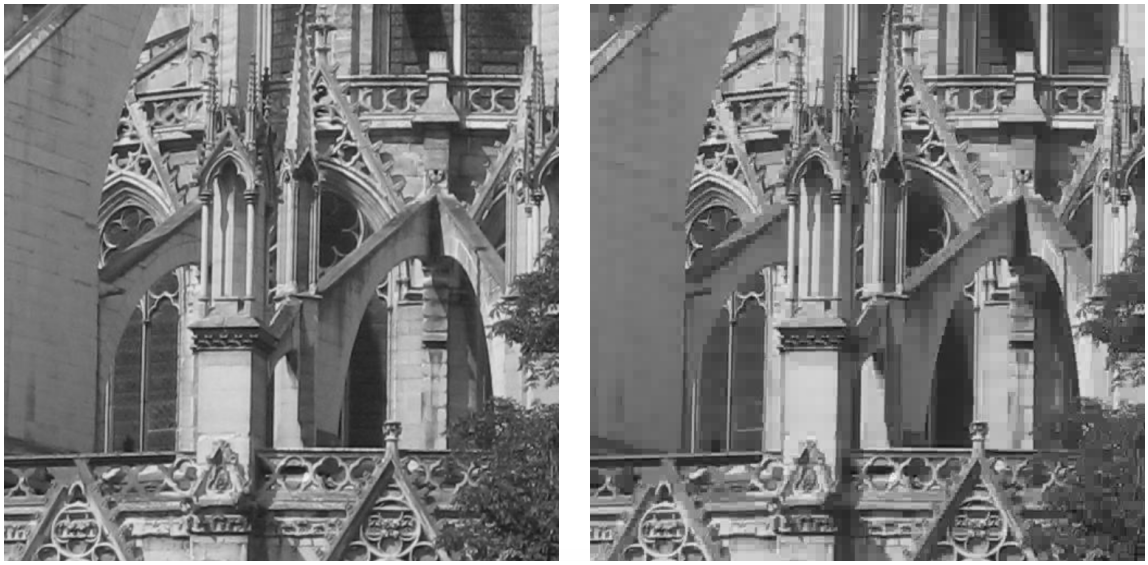


**Figure 3.4:** Original Image (left) | 30:1 Compression Ratio (right)

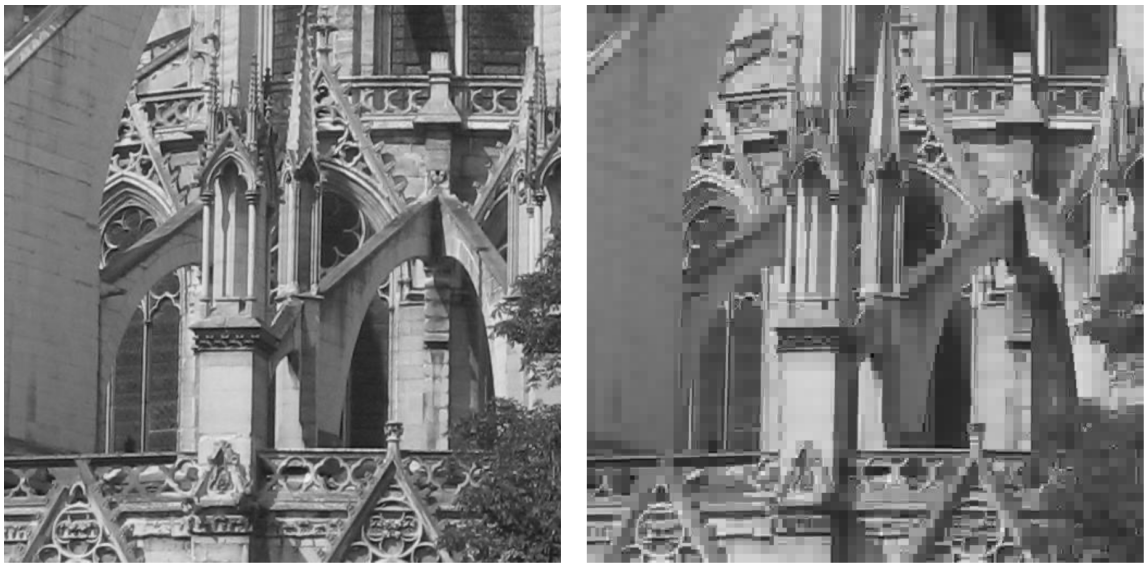
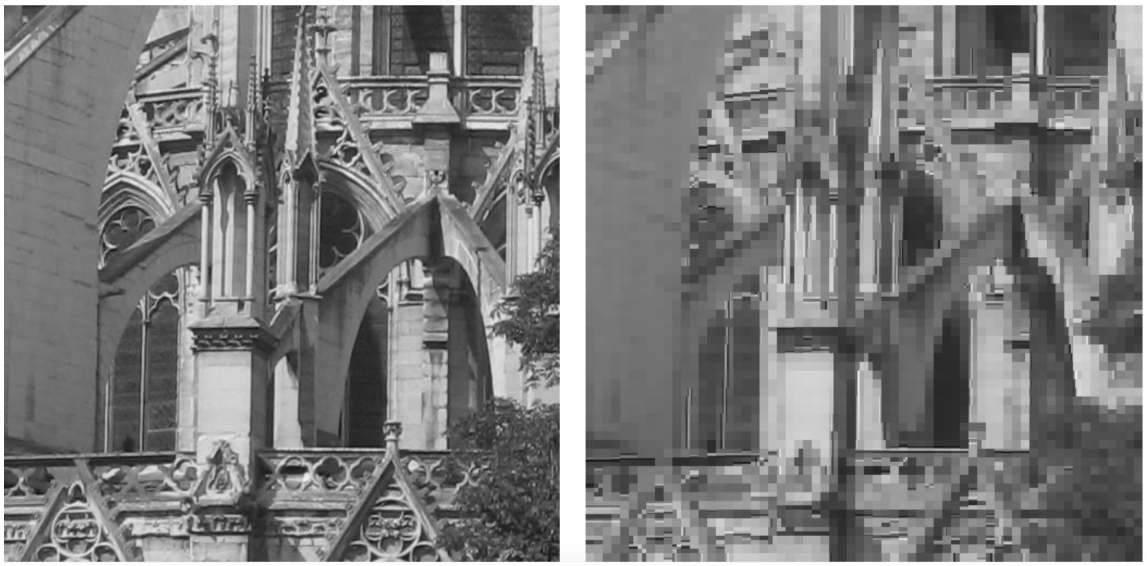


**Figure 3.5:** Original Image (left) | 50:1 Compression Ratio (right)



## 3.5 Normalization

Recall that a nxn square matrix A is said to be orthogonal if each of its columns forms an orthonormal basis for Rn, meaning that each column of A is pairwise orthogonal and each column vector has a length of 1. In other words, if the inverse of A equals the transpose, then A is orthogonal. This latter characteristic allows using the equation to retrieve the converted image easier and much faster.

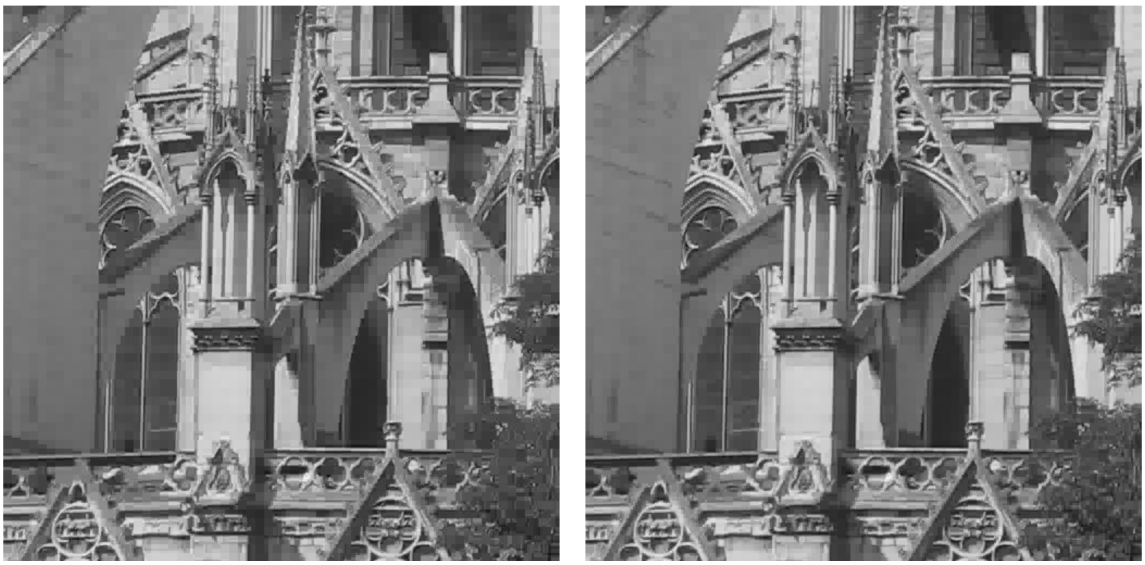
$$A = (HT)-1BH-1 = (H-1)T\ BH-1 = HBHT$$

If the angle between the two vectors Au and Av is, and A is an orthogonal matrix, then:

***cos(ψ) = ((Au)(Av))/(‖ Au ‖ ‖ Av ‖ ) = ((Au)T (Av))/(‖ u ‖ ‖ v ‖ ) = (utAT Av)/‖ u ‖ ‖ v ‖ = uT v/(‖ u ‖ ‖ v ‖ ) = (u.v)/(‖ u ‖ ‖ v ‖ ) = cos(ψ)***

When an orthogonal matrix is employed, there is substantially less distortion produced in the rebuilt image because both magnitude and angle are kept. One can normalize H by normalizing each of the three matrices because the transformation matrix H is the product of the three other matrices.

**Figure 3.6: Original Image (left) | Normalized, 10:1 Compression Ratio (right)**



## 3.6 Progressive Transmission

The progressive transmission of a compressed image can also be accomplished using the Haar wavelet transformation.

The source computer recalls the Haar converted matrix from memory each time you click on an image to download it from the Internet. The bigger detail coefficients and overall approximation coefficients are sent first, followed by the smaller detail coefficients. Your computer starts rebuilding the original image in ever more detail as it receives the information and continues until it is finished.

**Example:**

First, the Haar wavelet-compressed 50:1 image is delivered. This is decompressed by the receiving computer to display the 50:1 image. The only detailed coefficients transmitted after that are those that were canceled out in the 50:1 image but not in the 30:1 image. We can recreate the 30:1 image using this new information and the 50:1 image. The detail coefficients that were wiped out to produce the 30:1 image but not the 10:1 image are sent in a similar manner. We create the 10:1 image from this. In order to recreate the original image, the final detail coefficients are given. At any point, we can halt the process and view the compressed version of the image rather than the original.

# Chapter 4

## Conclusion

We can conserve a significant amount of computer memory, which can be put to better use, by using SVD and Wavelet algorithms. We have covered in detail in this project how the image compression is done using SVD and Wavelet. Image compression techniques, which in turn depend on linear algebra, are heavily used by everyone from our personal computers to the servers of all the major networking websites to save memory.

# Chapter 5

## Bibliography

[1] A.K. Jain, “Fundamentals of Digital Image Processing”(1989).

[2] David S. Watkins. “Fundamentals of Matrix Computations” (1991).

[3] James W. Demmel “Applied Numerical Linear Algebra”(1997).

[4] Lloyd Trefethen and David Bau “Numerical Linear Algebra“ (1997).

[5] Gilbert Strang “ MIT OCW 18.06sc Linear Algebra”

[6] Ulrike von Luxburg, University of Tübingen “Mathematics for Machine Learning”