# Lossless Huffman coding for image compression and decompression based on block and code book size using K-Means algorithm in spatial and frequency domain

**Ali Tariq Bhatti[1], Dr. Jung H. Kim[2]**
[1,2]Department of Electrical & Computer engineering
[1,2]North Carolina A&T State University, Greensboro NC USA
**Email:** [1]atbhatti@aggies.ncat.edu, alitariq.researcher.engineer@gmail.com, ali_tariq302@hotmail.com
[2]kim@ncat.edu

**Abstract:-** Images are basic source of information for almost all scenarios that degrades its quality both in visually and quantitatively way. It is one of the best techniques to apply in spatial and frequency filtering domain. Image compression using K-Means algorithm is an area characterized by need for extensive experimental work to establish the viability of proposed solutions to a given problem which is highly used in all applications like medical imaging, satellite imaging, and in optimization problems, etc. In this research paper, read an image of equal dimensional size (width and length) from MATLAB. Initialize and extract M-dimensional vectors or blocks from that image. However, initialize and design a code-book of size N for the compression. Quantize that image by using K-Means Algorithm to design a decode with table-lookup for reconstructing compressed image of different 8 scenarios. Compute the histogram equalization, discrete fourier transform, and fourier spectrum for the original and reconstructed image. Hence, perform that image compression in spatial filtering domain to distinguish it with the frequency filtering domain. To get in more detailed and experimental purposes, how this compressed reconstructed image pass through low pass and high pass filter from different techniques (Ideal, Gaussian, and Butterworth) in frequency filtering domain. With the help of vector quantizing K-Means clustering algorithm, evaluate and analyze the performance metrics (compression ratio, bit-rate, PSNR, MSE and SNR) for reconstructed compress image with different scenarios depending on size of block and code-book. Once finally, check the execution time, how fast it computes that compressed image in one of the best scenarios. The main aim of Lossless Huffman coding using block and codebook size for image compression in spatial and frequency domain is to convert the image to a form better that is suited for analysis to human by using K-Means algorithm. In this research paper, performance metrics also notifies from scenario (figure 57(c), when M=16 and N=50, if threshold=0.1) to perform lesser entropy and higher the average length for lossless Huffman coding on image compression using K-Means Algorithm. Finally, in this research paper that scenario 8 from figure 3, image has a higher PSNR, SNR and lesser Compression ratio shows a better quality of reconstructed image using K-Means algorithm in terms of Lossless Huffman coding in spatial and frequency domain.

**Keywords:-** Image compression, K-Means algorithm, Huffman coding, Spatial Domain Filtering, Frequency Domain Filtering, High Pass Filter, Low Pass Filter, Bit Rate, Compression Ratio, SNR, PSNR, MSE.

## 1. INTRODUCTION

### 1.1 Image Compression

Now-a-days, image compression techniques are very common in a wide area of researches. There are two types of image compression which are lossless and lossy compression. Lossless compression has the ability to reconstruct the original image after compression is exact. and it is error-free compression that is more than 2:1 ratio. In lossy compression, the ratio can be obtained with some error between the original image and the reconstructed image. In addition, error-free reconstruction of the original image may be impossible in many cases for lossy compression. Consequently, lossy compression may produce an acceptable error that does not affect too much to the original image. This can be seen in fast transmission of still images over the Internet where the amount of error can be acceptable [12] [18].

### 1.2 Image Enhancement:

Image enhancement is among the simplest and most appealing areas of image compression. Image contrast enhancement [7] and [8] is a classical problem in image processing and computer vision. Image enhancement is considered as a preprocessing step in many areas like video/image processing applications [16] and [17] speech recognition, texture synthesis etc.

### 1.3 Quantization

**Quantization** is the most general idea in **Lossy** compression. It is an irreversible process and source of information loss. It is a process of representing a large infinite set of values with a much smaller set. The design of the quantization has a significant impact on the amount of compression obtained and loss incurred in a lossy compression scheme. There are two types of Quantization which are Scalar Quantization and Vector Quantization. Quantization can do two types of mapping which are Encoder Mapping and Decoder Mapping.

Quantization error is defined as: q(X)=X-Q(X)

**(a) Scalar Quantization**

**Scalar Quantization:** It is that type of quantization which maps an input value 'x' into a finite number of output values 'y'. In **Scalar Quantization**, each source output is quantized individually. Scalar Quantization is divided into (a) **Uniform Quantization,** which has equal quantization region and (b)**Non-uniform Quantization,** which has various length quantization region. Uniform quantization is simplest, most popular, and conceptually of great importance.

**(b) Vector Quantization:**

It is that type of quantization in which the blocks of source output are quantized. It based on principle of block coding and performs better than the Scalar Quantization. It is not widely used because of the difficulty in hardware implementation. It is a fixed to fixed length algorithm. Vector quantization results in a lower distortion than Scalar Quantization for a given rate. When the source output is correlated, vectors of source output values will tend to fall in clusters. A vector quantization is composed of two operations which is the encoder, and the decoder. Image, voice compression, and voice recognition are the applications commonly for Vector Quantization.

### 1.4 K-Means Algorithm:

K-Means Algorithm is the Clustering algorithm that follows a simple way to classify a given data set through a certain number of clusters. The main idea behind K-Means Algorithm is to define 'K' centroids in K-Means algorithm, one for each cluster. These centroids should be placed in the best way so they are much as possible far away from each other. One of the disadvantages of K-Means Algorithm is to ignore measurement errors, or uncertainty, associated with the data and it is also known as Error based Clustering.

Quantization is the process of limiting real numbers to discrete integer values. Vector quantization is a lossy compression technique based on block coding. It maps a vector to a codeword drawn from a predesigned codebook with the goal of minimizing distortion. K-Means is an unsupervised machine learning technique. The basic idea of the K-means cluster is to place N data points in an l-dimensional space into K clusters.

### 1.5 Histogram and Histogram Equalization

Histogram is introduced by Karl Pearson that explains the graphical representation of the distribution of numerical data based on continuous variables (quantitative variables). Eventually, the histogram maps luminance, which is defined from the way the human eye perceives the brightness of different colors and therefore every pixel in the Color or Gray image computes to a luminance value between 0 and 255.

Histogram equalization (HE) is that type of equalization which is used for adjusting image intensities to enhance contrast. Contrast enhancement problem in digital images can be resolved using various methodologies, but (HE) [15] technique is the widely used one. Hence, this technique flattens the histogram and stretches the dynamic range of intensity values by using the cumulative density function. However, there are major draw backs in Histogram Equalization [10] especially when implemented to process digital images.

### 1.6 Spatial and Frequency domain filtering methods

Enhancement techniques mainly fall into two broad categories: spatial domain methods and frequency domain methods [15].

Spatial domain techniques are more popular than the frequency domain methods because they are based on direct manipulation of pixels in an image such as logarithmic transforms, power law transforms, and histogram equalization. However, these pixel values are manipulated to achieve desired enhancement. But they usually enhance the whole image in a uniform manner which in many cases produces undesirable results [5].

As, Frequency domain methods are based on the manipulation of the orthogonal transform of the image rather than the image itself. Frequency or Transformation domain techniques are suited for processing the image according to the frequency content [1]. Therefore, orthogonal transform of the image has two components which one are the magnitude and phase. The magnitude consists of the frequency content of the image. The phase is used to restore the image back to the spatial domain [5].

Basically, the idea behind enhancement techniques is to bring out detail that is obscured [3]. When pictures are converted from one form to another by processes such as imaging, scanning, or transmitting, the quality of the output image may be inferior to that of the original input picture [2].

### 1.7 Discrete Fourier transform and Fourier Spectrum in Frequency Domain

The image f(x,y) of size of block 'M' x size of codebook 'N' will be represented in the frequency domain (F(U,V)). 'U' and 'V' are the sine and cosine variables. The equation for the two-dimensional Discrete Fourier transform (DFT) is:

$$f(U,V) = \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f(x,y) e^{-j2\pi(\frac{Ux}{M}+\frac{Vy}{N})} \quad (1)$$

The Fourier transform can be constructed using a sum of sine and cosine waves of different frequencies. The exponential from the above equation can be expanded into sines and cosines with the variables U and V determining these frequencies. The inverse of the above discrete Fourier transform is as:

$$f(x,y) = \frac{1}{MN} \sum_{U=0}^{M-1} \sum_{V=0}^{N-1} f(U,V) e^{j2\pi(\frac{Ux}{M}+\frac{Vy}{N})} \quad (2)$$

Once, having F(U,V), compute the corresponding image (f(x,y)) using the Inverse, Discrete Fourier transform. Consequently, the value of the Inverse Discrete Fourier

transform at the origin of the frequency domain, at F(0,0), is called the DC component. However, F(0,0) is equal to MN times the average value of f(x,y). In addition, in MATLAB, F(0,0) is actually F(1,1) because array indices in MATLAB start at 1 rather than 0. The values of the Fourier transform are complex having real and imaginary parts. The imaginary parts are represented by i, which is the square root of -1. Analyze a Fourier transform by computing a **Fourier spectrum** in terms of the magnitude of F(U,V). Therefore, display it as an image where the Fourier spectrum is symmetric about the origin.

### 1.8 Discrete Fourier transform and Fourier spectrum in Spatial Domain

Thus, convolution theorem shows relationship between the spatial and frequency domain for the original and reconstructed image

$$f(x,y)*h(x,y)=H(U,V)F(U,V) \quad (3)$$

Therefore, $f(x,y)h(x,y) \Leftrightarrow H(U,V)*G(U,V)$ (4)

From the equation, "*" indicates convolution of the two functions that is the multiplication of two Fourier transforms which corresponds the convolution of the associated functions in the spatial domain.

### 1.9 Low and High pass filters in frequency domain

There are two commonly filters in the frequency domain which are explained in this research paper are as:

(a) Low pass filters: These filters are also known as Smoothing filters. However, these filters create a blurred or smoothed image. They attenuate the high frequencies and leave the low frequencies of the Fourier transform relatively unchanged.

(b) High pass filters: These filters are also known as Sharpening filters. However, this filter sharpened or represents the details of edge and noise of an image. They attenuate the low frequencies and leave the high frequencies of the Fourier transform relatively unchanged.

### 1.10 Performance Metrics:

There are following performance metrics used for image compression of original and reconstructed image such as

**(a) Bit Rate:**

Bit Rate is defined as

$$\text{Bit Rate} = \frac{\text{Number of bits to index a vector}}{\text{Number of samples in a vector}} \quad (5)$$

$$\text{Bit Rate} = \frac{\log_2 N}{(M)} \quad (6)$$

The units for Bit Rate is bits/pixel.

**(b) Compression Ratio:**

Compression Ratio is defined as:

$$\text{Compression Ratio} = \frac{\text{Original Bit Rate}}{\text{New Bit Rate}} \quad (7)$$

Compression Ratio is Unit-less.

**(c) SNR:**

**SNR (Signal-To-Noise Ratio) is defined as**

$$SNR = \frac{10\log_{10}*(\sum^{n}_{i=0} X_i^2)}{(\sum^{n}_{i=0} (Y_i - X_i)^2)} \quad (8)$$

**(d) MSE:**

The Mean Square Error (MSE) is the error metric used to compare image quality. The MSE represents the cumulative squared error between the reconstructed($\mathbf{Y}_i$) and the original image($X_i$). MSE is a risk function corresponding to the expected value of the squared error loss or quadratic loss. Mean Square Error (MSE) is given by

$$MSE = \frac{sum(sum((Y_i - X_i)^2))}{(M * N)} \quad (9)$$

**(e) PSNR**

Peak Signal-to-Noise Ratio short as PSNR, is an engineering term for the ratio between the maximum possible power of a signal and the power of corrupting noise that affects the fidelity of its MSE representation. PSNR is usually expressed in terms of the logarithmic decibel scale.

$$PSNR = \frac{10*\log_{10}(256^2))}{MSE} \quad (10)$$

To this end, image compression is an important research issue [4]. Consequently, images contain large amounts of information that requires much storage space, large transmission bandwidths and long transmission times. Moreover, it is beneficial to compress the image by storing only the essential information needed to reconstruct the image.

## 2. Block Diagram Implementation

The block diagram explained as

Step 1: Read an image of 256X256 with it equal dimensions Convert RGB to Gray scale level.

Step2: Initialization of the size of block 'M' and size of codebook 'N' for different scenarios

Step3: Quantizing K- Mean clustering for an image

There are 4 cases to use K-Mean Algorithm, which are as:

(a) Initialize a set of training vectors with any variable as 'X' and we need a codebook of size N as in this case.
(b) Second case is to randomly choose M dimensional or block vectors as the initial set of code words in the codebook.
(c) Third case is to search for nearest neighbor for each training vector. This will allow finding the codeword in the current codebook which seems to be closest in terms of spectral distance and assign that vector to the corresponding cell.
(d) Finally update the Centroid for the code word in each cell using the training vectors assigned to that cell. In this case 4, repeat case 2 and 3 again and again until the procedure converges or Average distance falls below a preset threshold.
Step 4: Compute the histogram for the original and reconstructed image
Step 5: Compute spatial filtering of the reconstructed image. Also compute frequency filtering domain of the reconstructed image to pass it through low-pass and high-pass filter
Step 6: Compute discrete Fourier transform and Fourier spectrum of the reconstructed image.
Step 7: Compute Huffman coding quad-tree decomposition by encoding the block and codebook of image using K-Mean algorithm to compress it. Then decode to decompress it.
Step 8: Implementation of performance metrics such as Bit rate, Compression ratio, SNR, MSE, and PSNR for reconstructed based on one of the following scenarios.
Step 9: Check the execution time of one the scenarios for reconstructed image depending on the size of block and codebook.

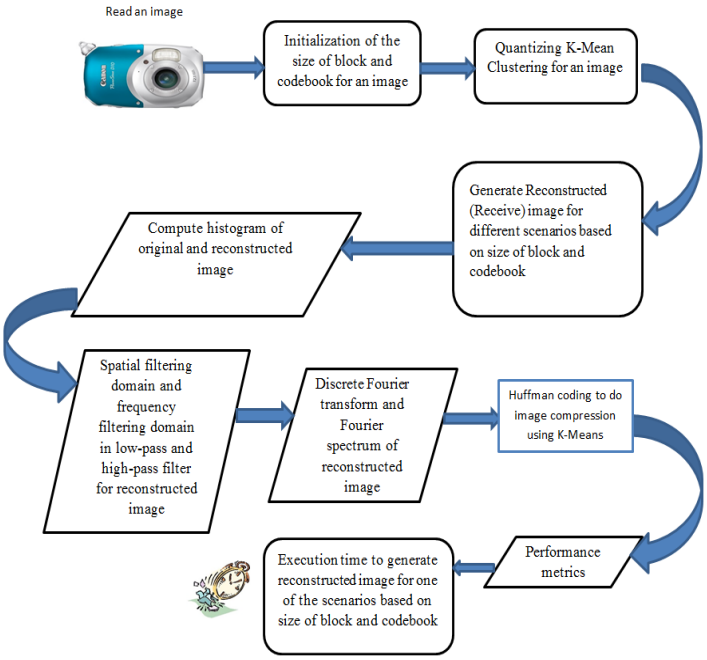


**Figure 1 Block diagram of lossless Huffman coding Image Compression using K-Mean Algorithm and other computed aspects**

## 3. ORIGINAL AND RECONSTRUCTED IMAGES USING K-MEANS ALGORITHM OF 8 DIFFERENT SCENARIOS

The original image size is 256X256 read from MATLAB in figure 2, and obtained the reconstructed image (figure 3 through figure 10) for different 8 scenarios using K-Mean algorithms.


**Figure 2 Original Image**

Scenario#8 Size of Block=M=16, and Size of Codebook=N=50 (**16X50)**


**Figure 3 Reconstructed Image of 16X50**

Scenario#7 Size of Block=M=16, and Size of Codebook=N=25 **(16X25)**


**Figure 4 Reconstructed Image of 16X25**

Scenario#6 Size of Block=M=64, and Size of Codebook=N=50 **(64X50)**


**Figure 5 Reconstructed Image of 64X50**

Scenario#5 Size of Block=M=64, and Size of Codebook=N=25 **(64X25)**

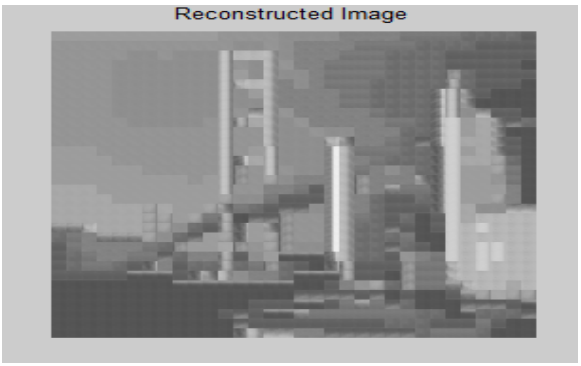


**Figure 6 Reconstructed Image of 64X25**

Scenario#4 Size of Block=M=256, and Size of Codebook=N=50 **(256X50)**


**Figure 7 Reconstructed Image of 256X50**

Scenario#3 Size of Block=M=256, and Size of Codebook=N=25 **(256X25)**


**Figure 8 Reconstructed Image of 256X25**

Scenario#2 Size of Block=M=1024, and Size of Codebook=N=50 **(1024X50)**


**Figure 9 Reconstructed Image of 1024X50**

Scenario#1 Size of Block=M=1024, and Size of Codebook=N=25 **(1024X25)**


**Figure 10 Reconstructed Image of 1024X25**

## 4. Histogram Equalization of Original and Reconstructed Image

Histogram Equalization enhances the contrast of images by transforming the values in an intensity image so that the histogram of the output image is approximately flat. Histogram of original image from figure 2 and reconstructed image from figure 3 are shown in figure 11 and figure 12.

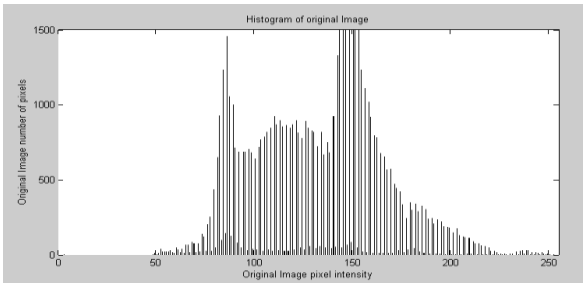


**Figure 11 Histogram of Original image**

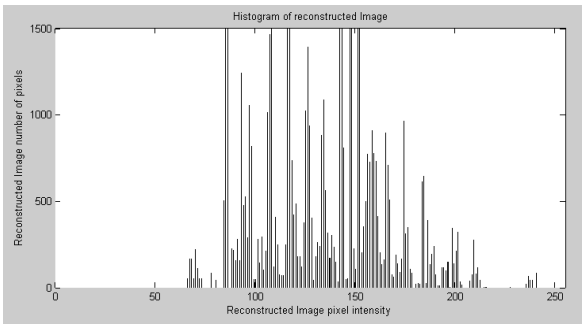


**Figure 12 Histogram of Reconstructed image**

## 5. Spatial Domain Filtering for Original and Reconstructed Image

In this figure 13, Spatial domain filtering for original image from figure 2 is obtained by K-Means algorithm. Apply the fspecial function('sobel') function that generates a 3-by-3 filter to emphasizes horizontal edges, and should transpose it to emphasize vertical edges. Use the imfilter function to perform image filtering using convolution of having negative values as shown in figure 13.


**Figure 13 Spatial Domain filtering of Original image having negative values**

In this figure 14, Spatial domain filtering for reconstructed image from figure 3 is obtained by K-Means algorithm.

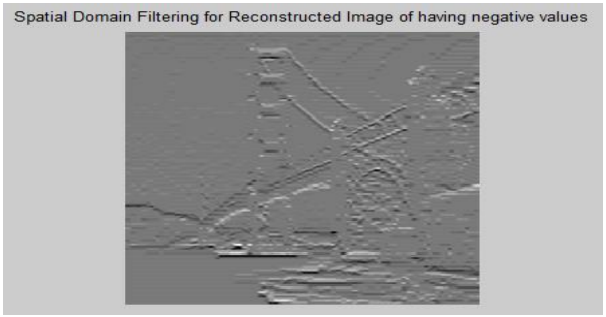


**Figure 14 Spatial Domain filtering of Reconstructed image having negative values**

In this figure 15, Spatial domain filtering for original image from figure 2 is obtained by K-Means algorithm.

Apply the fspecial function('sobel') function that generates a 3-by-3 filter to emphasizes horizontal edges, and should transpose it to emphasize vertical edges. Use the imfilter function to perform image filtering using convolution. Therefore, then applied the absolute function to get rid of negatives as shown in figure 15.

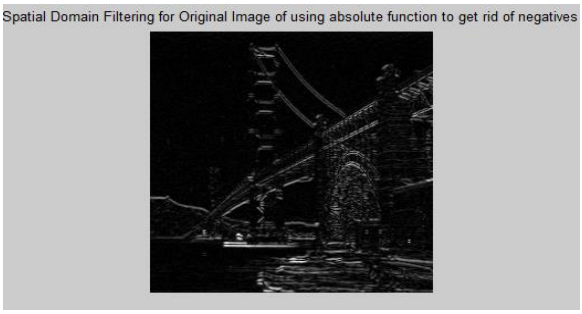


**Figure 15 Spatial Domain filtering of Original image using absolute values**

In this figure 16, Spatial domain filtering for reconstructed image from figure 3 is obtained by K-Means algorithm.

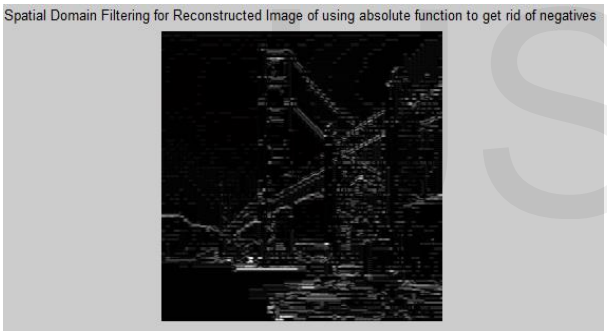


**Figure 16 Spatial Domain filtering of Reconstructed image using absolute values**

In this figure 17, Spatial domain filtering for original image from figure 2 is obtained by K-Means algorithm. Apply the fspecial function('sobel') function that generates a 3-by-3 filter to emphasizes horizontal edges, and should transpose it to emphasize vertical edges. Use the imfilter function to perform image filtering using convolution. Therefore, then applied the absolute function that is greater than the maximum of absolute reconstructed image filtering having threshold of 0.2 into a binary image as shown in figure 17.


**Figure 17 Spatial Domain filtering of Original image having threshold into a binary image**

In this figure 18, Spatial domain filtering for reconstructed image from figure 3 is obtained by K-Means algorithm.


**Figure 18 Spatial Domain filtering of Reconstructed image having threshold into a binary image**

## 6. Frequency Domain Filtering for Original and Reconstructed Image

The first definition of a quaternion Fourier transform was that of [11] and the first application of a quaternion Fourier transform of color images was reported in 1996 [9] using a discrete version of Ell' transform. In this work, we use the more recent quaternion Fourier transform definition [6]. The application of a quaternion Fourier transform to colour images is based on representing color image pixels using quaternions discovered by Hamilton in 1843 [4].

In this figure 19, Frequency domain filtering for original image from figure 2 is obtained by K-Means algorithm. Apply the fspecial function('sobel') function to do fft2(fast fourier transform for 2-dimensional image) that generates a 3-by-3 filter to emphasizes horizontal edges, and should transpose it to emphasize vertical edges. Use the padded size function to perform fft2(fast fourier transform for 2 dimensional image) filtering. Then, multiply both results which we performed for fft2 and therefore, then finally perform ifft2(inverse fast fourier transform for 2 dimensional image) to obtain negative values as shown in figure 19.

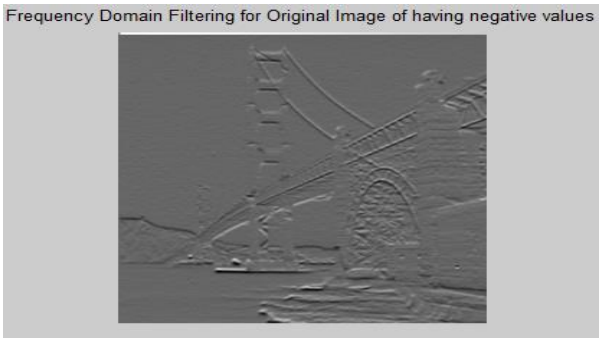


**Figure 19 Frequency Domain filtering of Original image having negative values**

In this figure 20, Frequency domain filtering for reconstructed image from figure 3 is obtained by K-Means algorithm.

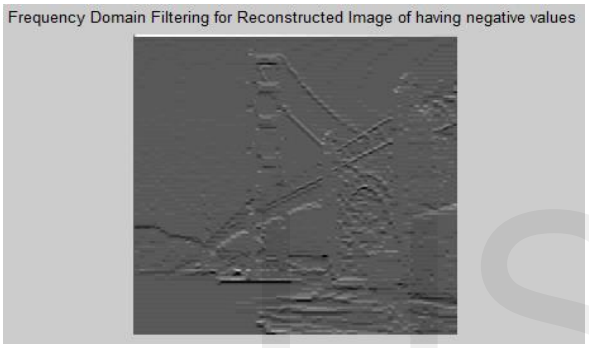


**Figure 20 Frequency Domain filtering of Reconstructed image having negative values**

In this figure 21, Frequency domain filtering for original image from figure 2 is obtained by K-Means algorithm. Apply the fspecial function('sobel') function to do fft2(fast fourier transform for 2-dimensional image) that generates a 3-by-3 filter to emphasizes horizontal edges, and should transpose it to emphasize vertical edges. Use the padded size function to perform fft2(fast fourier transform for 2 dimensional image) filtering. Then, multiply both results which we performed for fft2 and therefore, then finally perform ifft2(inverse fast fourier transform for 2 dimensional image). Therefore, then applied the absolute function to get rid of negatives as shown in figure 21.

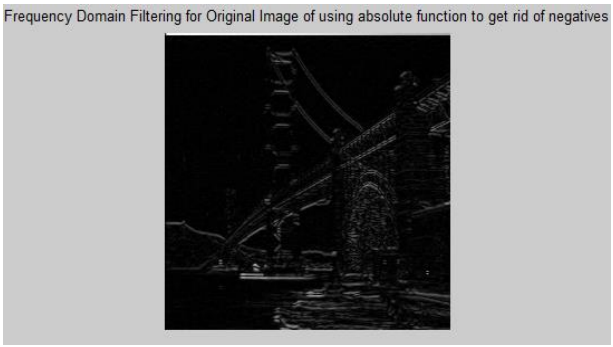


**Figure 21 Frequency Domain filtering of Original image using absolute values**

In this figure 22, Frequency domain filtering for reconstructed image from figure 3 is obtained by K-Means algorithm.

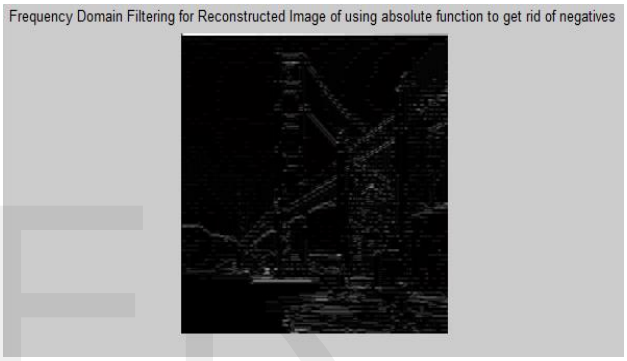


**Figure 22 Frequency Domain filtering of Reconstructed image using absolute values**

In this figure 23, Frequency domain filtering for original image from figure 2 is obtained by K-Means algorithm. Apply the fspecial function ('sobel') function to do fft2(fast fourier transform for 2-dimensional image) that generates a 3-by-3 filter to emphasizes horizontal edges, and should transpose it to emphasize vertical edges. Use the padded size function to perform fft2(fast fourier transform for 2 dimensional image) filtering. Then, multiply both results which we performed for fft2 and therefore, then finally perform ifft2(inverse fast fourier transform for 2 dimensional image). Therefore, then applied the absolute function that is greater than the maximum of absolute reconstructed image having threshold of 0.2 into a binary image as shown in figure 23 for original image.


**Figure 23 Frequency Domain filtering of Original image having threshold into a binary image**

In this figure 24, Frequency domain filtering for reconstructed image from figure 3 is obtained by K-Means algorithm.


**Figure 24 Frequency Domain filtering of Reconstructed image having threshold into a binary image**

### 6.1 Frequency Domain Filtering in low pass filter using Discrete Fourier and Fourier Spectrum transform

Calculate the low pass filter padded sizes to obtain FFT-based filtering transform using MATLAB which is a fast algorithm. Generate a filter function and multiply it with the transform. Obtain the real part of the inverse FFT. Crop the top, left rectangle to the original(figure 2) and reconstructed(figure 3) image size to obtain Discrete Fourier transform as shown in figure 25 and figure 26.


**Figure 25 Low Pass Filter using Discrete Fourier transform for Original image**


**Figure 26 Low Pass Filter using Discrete Fourier transform for Reconstructed image**

To compute the fourier spectrum of low pass filter original(figure 2) and reconstructed(figure 3) image, move the origin of the transform to the center of the frequency rectangle. Then, use absolute (abs) to compute the magnitude of imaginary part and use log to brighten display as shown in figure 27 and figure 28.

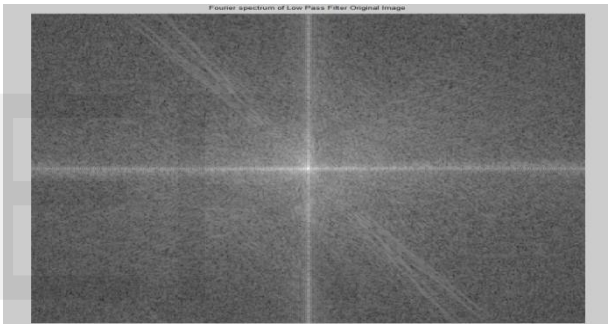


**Figure 27 Low Pass Filter using Fourier Spectrum for Original image**

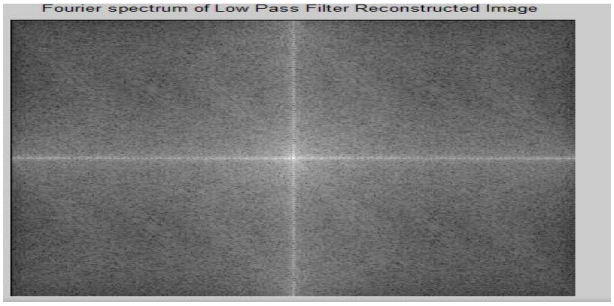


**Figure 28 Low Pass Filter using Fourier Spectrum for Reconstructed image**

### 6.2 Frequency Domain Filtering using three types of low pass filter

From section 1.8 equations, multiplying the fast fourier transforms of two functions from the spatial domain produces the convolution of those functions, so use Fourier transforms as a fast convolution on large images. As a note, on small images, it is faster to work in

the spatial domain. However, we can also create filters directly in the frequency domain.

Three main low pass filters implemented in MATLAB in this research paper are

(a)Ideal low pass filter (ILPF)

(b)Gaussian low pass filter (GLPF)

(c)Butterworth low pass filter (BLPF)

Therefore, the corresponding equations and visual representations of these filters are shown below. In the equation (11-13), $D_0$ is a specified non-negative number. D(U,V) is the distance from point (U,V) to the center of the filter.

**(a) Ideal Low Pass Filter (ILPF)**

Ideal low pass filter is a simple cutoff all the high frequency ranges those are higher than the specified cutoff frequency. From the figure below, these low pass filter cutoff all high frequency components of the Fourier transform that are at the distance greater than distance $D_0$ from the center.

From equation (11), $D_0$ is a cutoff frequency and D(U,V) is the distance from point (U,V) to the center of the frequency rectangle. The equation is as:

$$H(U,V)= \{ 1 \quad \text{if } D(U,V) \leq D_0$$
$$0 \quad \text{if } D(U,V) > D_0 \ \} \qquad \textbf{(11)}$$

Hence, these type of filters are radially symmetric about the origin, which means that the filter is completely defined by radial cross section. Unlike the ILPF, the BLPF transfer function does not have a sharp discontinuity that gives a clear cutoff between passed and filtered as shown in figure 29 and figure 30 for original(figure 2) and reconstructed(figure 3) image.

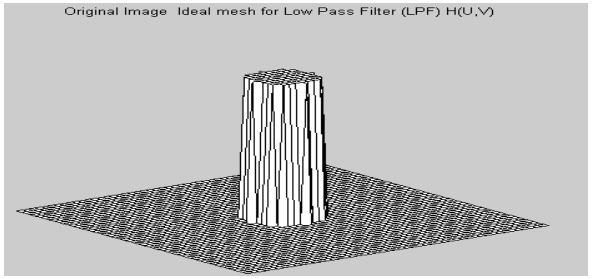


**Figure 29 Ideal mesh Low Pass Filter for Original image in 3D plot**

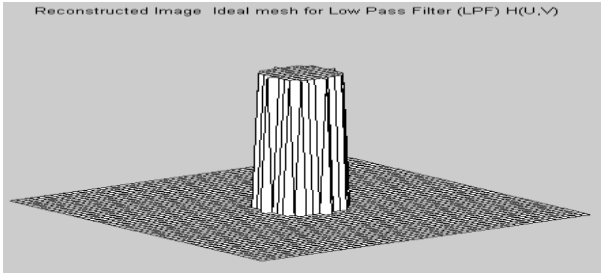


**Figure 30 Ideal mesh Low Pass Filter for Reconstructed image in 3D plot**

The cutoff frequency increases, then blurring effect is decrease and then the ringing effect become finer as in the figure 31 and figure 32 for original(figure 2) and reconstructed(figure 3) image.

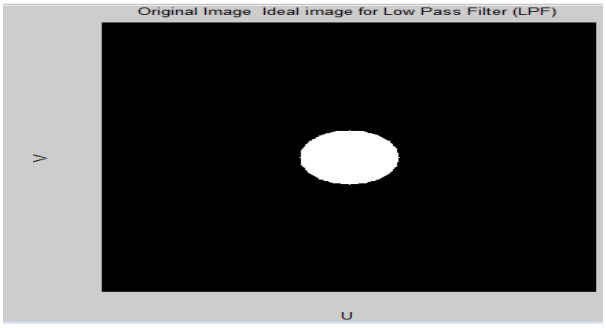


**Figure 31 Ideal Low Pass Filter for Original image**

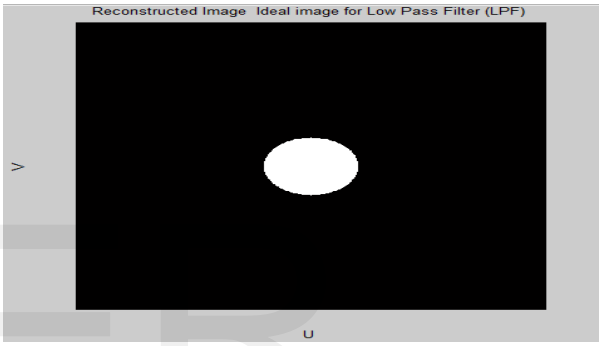


**Figure 32 Ideal Low Pass Filter for Reconstructed image**

**(b) Gaussian Low Pass Filter (GLPF)**

This type of low pass filter transfer function does not have a sharp transition, but it has a smooth transition between high and low frequencies. It can be used for character recognized. They are unable to achieve as much smoothing as the Butterworth Low Pass Filter of order 2 for the same value of cutoff frequency. The equation is as:

$$H(U,V) = e^{-D^2(U,V)/2D_0^2} \qquad (12)$$

Unlike the ILPF, GLPF transfer function also does not have sharp discontinuity but it is much smoother than BLPF. However, Butterworth filter represents the transition between the sharpness of the ideal filter and broad smoothness of the Gaussian filter as shown in figure 33 and figure 34 for original(figure 2) and reconstructed(figure 3).

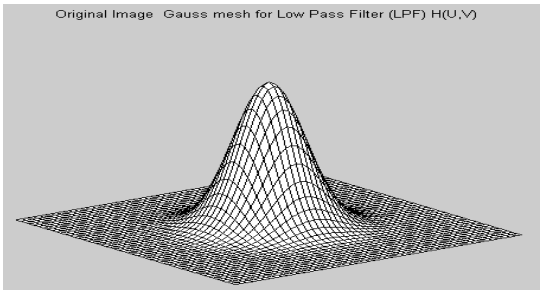


**Figure 33 Gaussian mesh Low Pass Filter for Original image in 3D plot**

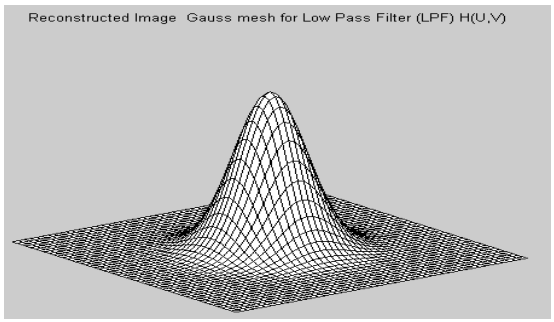


**Figure 34 Gaussian mesh Low Pass Filter for Reconstructed image in 3D plot**

When the cutoff frequency is increased, smooth transition tends towards blurring as in figure 35 and figure 36 for original(figure 2) and reconstructed image(figure 3).

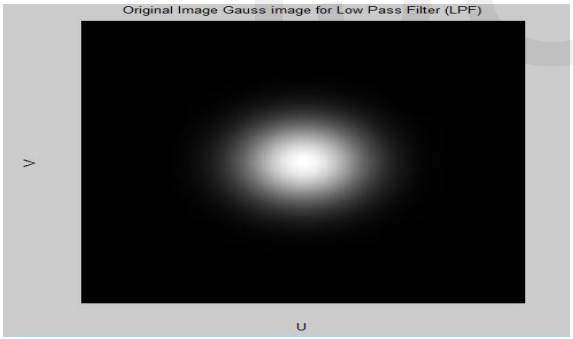


**Figure 35 Gaussian Low Pass Filter for Original image**

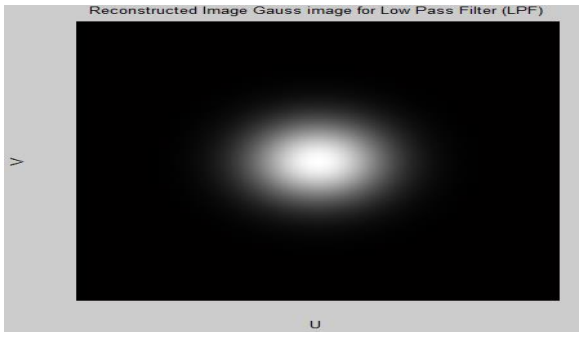


**Figure 36 Gaussian Low Pass Filter for Reconstructed image**

**(c) Butterworth Low Pass Filter (BLPF)**

It is a simple low pass filter that cutoff all the high frequencies higher than the specific cutoff frequency as shown in figure 37 and figure 38 for original(figure 2) and reconstructed(figure 3) image. From equation (13), they are of order n, and with cutoff frequency at a distance $D_0$ from the center. The equation is as

$$H(U,V) = \frac{1}{1+[\frac{D(U,V)}{D_0}]^{2n}} \qquad (13)$$

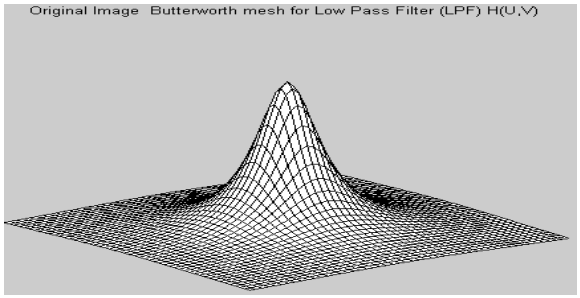


**Figure 37 Butterworth mesh Low Pass Filter for Original image in 3D plot**

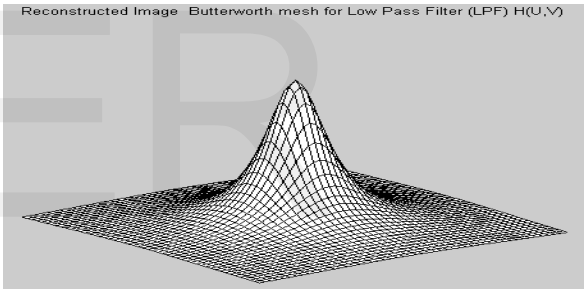


**Figure 38 Butterworth mesh Low Pass Filter for Reconstructed image in 3D plot**

These filters with order 1 have no any ringing effect. Therefore, these filters with order 2 or more increased ringing effect as the order is increased shown in figure 39 and figure 40 for original(figure 2) and reconstructed(figure 3) image.

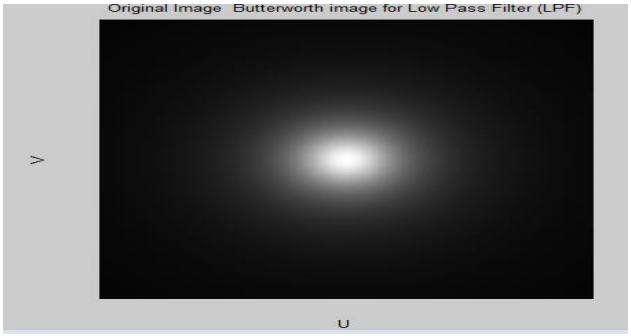


**Figure 39 Butterworth Low Pass Filter for Original image**

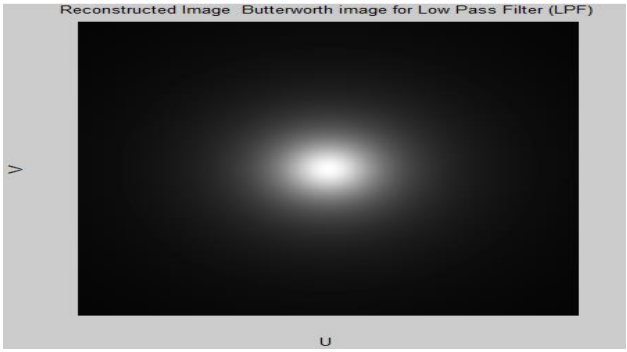


**Figure 40 Butterworth Low Pass Filter for Reconstructed image**

### 6.3 Frequency Domain Filtering in high pass filter using Discrete Fourier transform and Fourier spectrum

Calculate the high pass filter padded sizes to obtain FFT-based filtering transform using MATLAB which is a fast algorithm. It is applied from original(figure 2) and reconstructed(figure 3) as shown in figure 41 and figure 42.


**Figure 41 High Pass Filter using Discrete Fourier transform for Original image**


**Figure 42 High Pass Filter using Discrete Fourier transform for Reconstructed image**

To compute the fourier spectrum of high pass filter original for original(figure 2) and reconstructed(figure 3) as shown in figure 43 and figure 44.

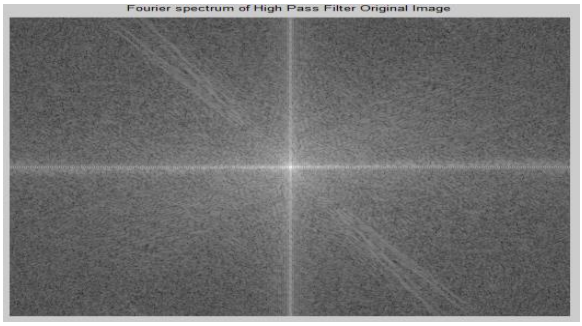


**Figure 43 High Pass Filter using Fourier Spectrum for Original image**

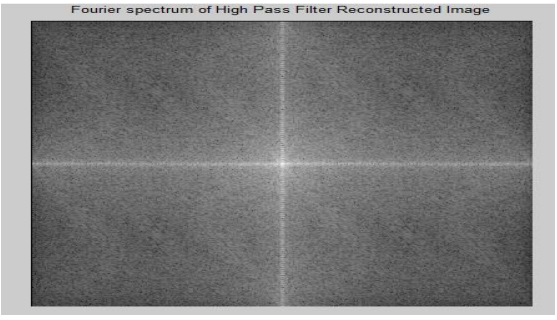


**Figure 44 High Pass Filter using Fourier Spectrum for Reconstructed image**

### 6.4 Frequency Domain Filtering using three types of high pass filter

Three main high pass filters implemented in MATLAB in this research paper are

(a)Ideal high pass filter (IHPF)

(b)Gaussian high pass filter (GHPF)

(c)Butterworth high pass filter (BHPF)

The highpass filter ($H_{highpass}$) is often represented by its relationship to the lowpass filter ($H_{lowpass}$):

$H_{highpass}(U,V) = 1 - H_{lowpass}(U,V)$

**(a) Ideal High Pass Filter (IHPF)**

It is a simple high pass filter which simply cutoff all the low frequency lower than the specified cutoff frequency. $D_0$ is a predefined nonnegative quantity corresponding to the filters cutoff frequency as shown in figure 35.

From equation (14), D(U,V) is a distance measure in the (shifted) Fourier spectrum. Ideal filters causes severe ringing effects for original(figure 2) and reconstructed(figure 3) image, so figure 45 and figure 46 shown as

$$H(U,V) = \begin{cases} 0 & \text{if } D(U,V) \le D_0 \\ 1 & \text{if } D(U,V) > D_0 \end{cases} \quad \textbf{(14)}$$

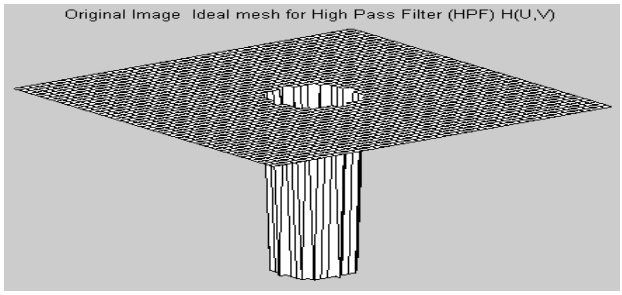


**Figure 45 Ideal mesh High Pass Filter for Original image in 3D plot**

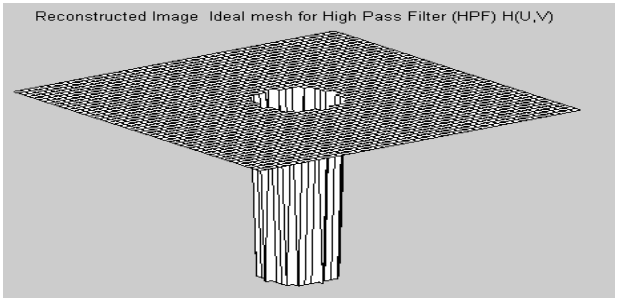


**Figure 46 Ideal mesh High Pass Filter for Reconstructed image in 3D plot**

When the cutoff frequency is increased, ringing effect and edge distortion effect are decreased for original(figure 2) and reconstructed(figure 3) image as shown in figure 47 and figure 48.

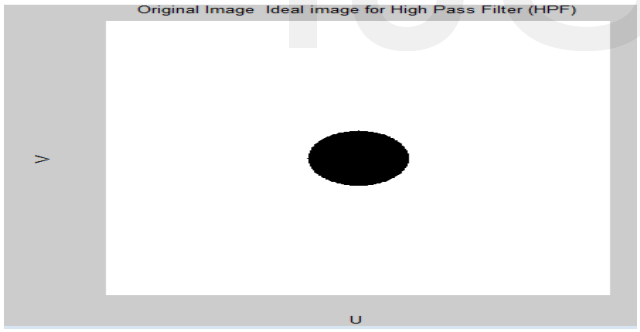


**Figure 47 Ideal High Pass Filter for Original image**

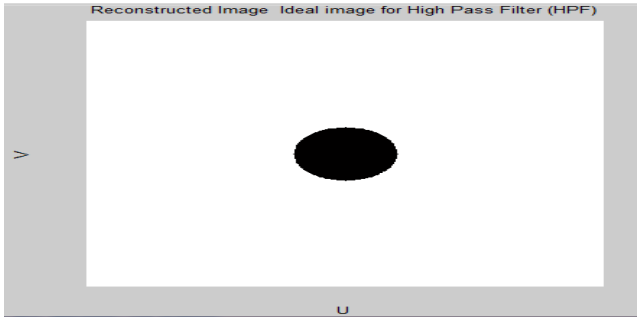


**Figure 48 Ideal High Pass Filter for Reconstructed image**

**(b) Gaussian High Pass Filter (GHPF)**

It is a simple high pass filter like other high pass filter that cutoff all the low frequency component. Smoother results are obtained and discard low frequencies as it keeps high frequencies.

Since Gaussian HP = (1 – Gaussian LP) does not introduce any ringing artifacts and can be created by varying the standard deviation $D_0$ to increase or decrease filter width. From equation (15), $D_0$ is the filters cutoff frequency and D(U,V) is a distance measure in the shifted Fourier spectrum. However, Gaussian filter causes no ringing effects.

$$H(U,V) = 1 - e^{D^2(U,V)/2D_0^2} \qquad (15)$$

By applying Butterworth High Pass Filter and GHPF on the images, it is found that the color intensities of high pass filter of fast fourier transform images drastically change. In fact, the split edges are elongated and wider in BHPF case than GHPF. But in GHPF case the split edges are sharper for original(figure 2) and reconstructed(figure 3) image as shown in figure 49 and figure 50. These elongated split edges in BHPF leads to sharper image than GHPF.

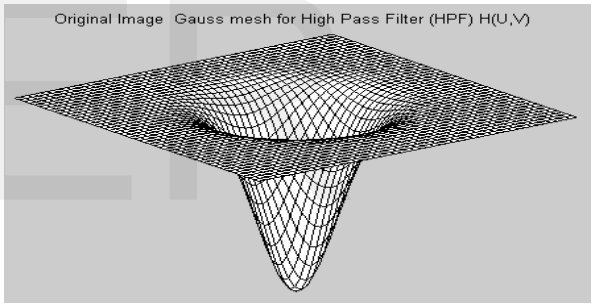


**Figure 49 Gaussian mesh High Pass Filter for Original image in 3D plot**

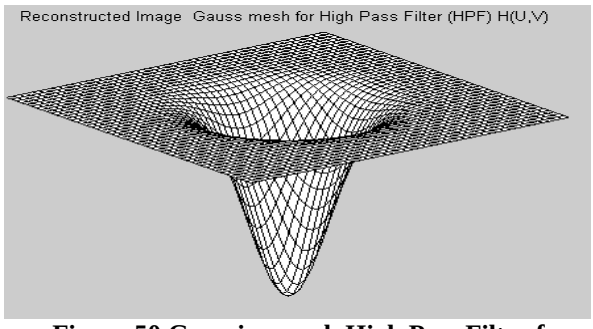


**Figure 50 Gaussian mesh High Pass Filter for Reconstructed image in 3D plot**

There is no ringing effect and Edge distortion is less for original(figure 2) and reconstructed(figure 3) image as shown in figure 51 and figure 52.

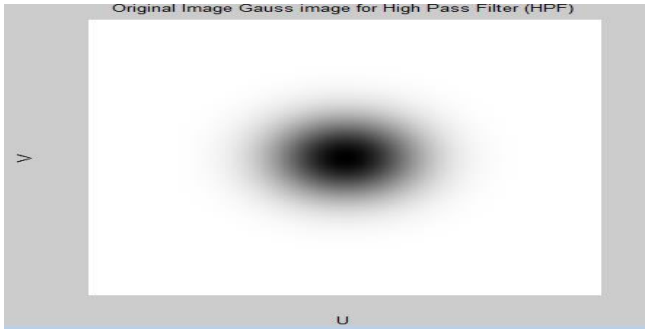


**Figure 51 Gaussian High Pass Filter for Original image**

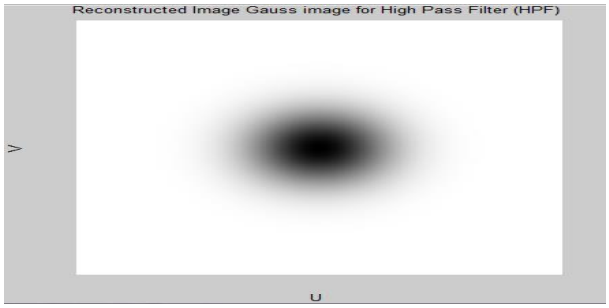


**Figure 52 Gaussian High Pass Filter for Reconstructed image**

**(c) Butterworth High Pass Filter (BHPF)**

It is a high pass filter, which cutoff all the low frequency components of an image. From equation (16), $D_0$ is the filters cutoff frequency. D(U,V) is a distance measure in the (shifted) Fourier spectrum. 'n 'is the order of the filter and determines the smoothness of the cutoff transition.

Higher order when n > 2, Butterworth filters causes increasing ringing effect for original(figure 2) and reconstructed(figure 3) image as shown in figure 53 and figure 54.

$$H(U,V) = \frac{1}{1+[\frac{D_0}{D(U,V)}]^{2n}} \qquad (16)$$

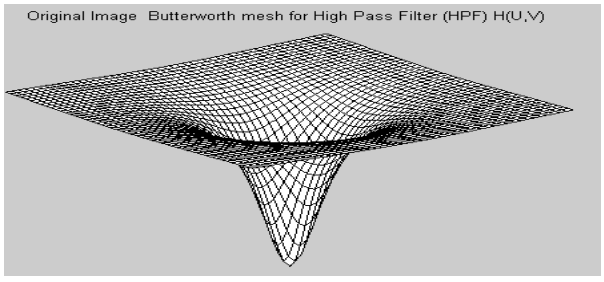


**Figure 53 Butterworth mesh High Pass Filter for Original image in 3D plot**

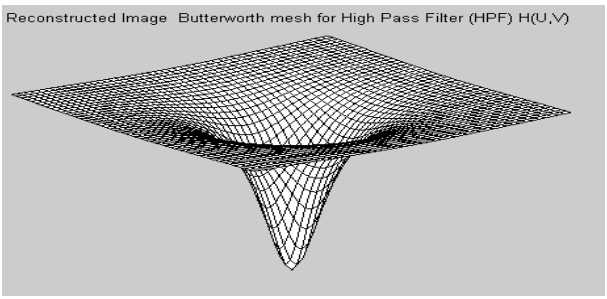


**Figure 54 Butterworth mesh High Pass Filter for Reconstructed image in 3D plot**

It gives smoother result compared to Ideal high pass filter for original(figure 2) and reconstructed(figure 3)image as shown in figure 55 and figure 56.

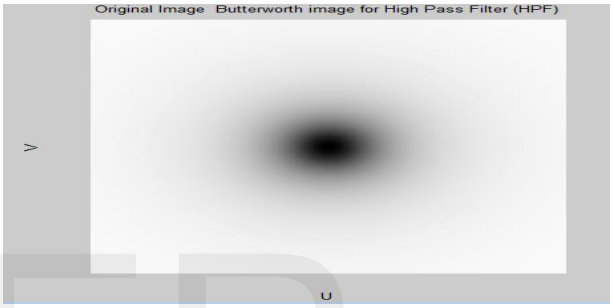


**Figure 55 Butterworth High Pass Filter for Original image**

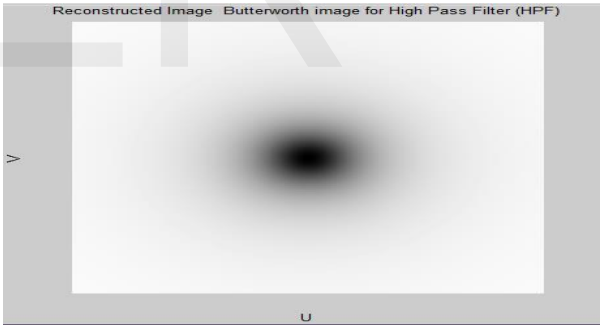


**Figure 56 Butterworth High Pass Filter for Reconstructed image**

## 7. COMPUTATIONAL RESULTS OF PERFORMANCE METRICS OF RECONSTRUCTED IMAGES OF 8 DIFFERENT SCENARIOS USING K-MEAN ALGORITHM

From the table and the results displayed in all 8 scenarios, M=16, and N=50 from figure 3 shows the better performance of reconstructed image using K-Means algorithm. In addition, it takes more execution time of 54.698798 sec than the other various scenarios. It shows more MSE and less PSNR. More the SNR is, the more is the Bit Rate and lesser the Compression ratio of 22.6795.

| Image size 256X256 | Bit Rate | Compression ratio | SNR | PSNR | MSE | Execution Time (sec) |
|---|---|---|---|---|---|---|
| M=1024 N=25 | 0.0045 | $1.7641*10^3$ | 20.0273 | 21.41 | 474.15 | 6.5541 |
| M=1024 N=50 | 0.0055 | $1.4515*10^3$ | 24.0779 | 28.47 | 93.29 | 8.8763 |
| M=256 N=25 | 0.0181 | 441.0128 | 19.8178 | 15.18 | 1990.31 | 8.4815 |
| M=256 N=50 | 0.0220 | 362.8725 | 21.1566 | 19.52 | 731.17 | 7.0436 |
| M=64 N=25 | 0.0726 | 110.2532 | 21.4792 | 10.82 | 5430.55 | 13.0607 |
| M=64 N=50 | 0.0882 | 90.7181 | 21.9804 | 14.33 | 2419.35 | 12.5786 |
| M=16 N=25 | 0.2902 | 27.5633 | 23.2873 | 6.60 | 14324.98 | 20.6180 |
| M=16 N=50 | 0.3527 | 22.6795 | 24.6071 | 10.93 | 5285.54 | 54.698798 |

**Table 1 Computational results of performance metrics of 8 different scenarios**

## 8. HUFFMAN CODING:

Lossless Huffman coding technique is applied on image compression using K-Means algorithm for 8 different scenarios.

**(i) The Quad tree decomposition**

This approach divides a square image into four equal sized square blocks, and then tests each block to see if meets some criterion of homogeneity. If a block size meets the criterion it is not divided any further, and the test criterion is applied to those blocks. This process is repeated iteratively until each block meets the criterion.

**(ii) Huffman Encoding**

The Huffman encoding starts by constructing a list of all the alphabet symbols in descending order of their probabilities. It then constructs, from the bottom up, a binary tree with a symbol at every leaf. This is done in steps, where at each step two symbols with the smallest probabilities are selected, added to the top of the partial tree, deleted from the list, and replaced with an auxiliary symbol representing the two original symbols [20]. When the list is reduced to just one auxiliary symbol (representing the entire alphabet), the tree is complete. The tree is then traversed to determine the code words of the symbols.

**(iii) Huffman Decoding**

Before starting the compression of an image using K-Means algorithm, the encoder has to determine the codes. It does that based on the probabilities of frequencies of occurrence of the symbols. The probabilities or frequencies have to be written, as side information, on the output, so that any Huffman decoder will be able to decompress the data. This is easy, because the frequencies are integers and the probabilities can be written as scaled integers. It normally adds just a few hundred bytes to the output. It is also possible to write the variable-length codes themselves on the output, but this may be awkward, because the codes have different sizes. It is also possible to write the Huffman tree on the output [19], but this may require more space than just the frequencies. In any case, the decoder must know what is at the start of the compressed file, read it, and construct the Huffman tree for the alphabet. Only then can it read and decode the rest of its input. The algorithm for decoding is simple. Start at the root and read the first bit off the input (the compressed image). If it is zero, follow the bottom edge of the tree; if it is one, follow the top edge. Read the next bit and move another edge toward the leaves of the tree. When the decoder arrives at a leaf, it finds there the original, uncompressed symbol, and that code is emitted by the decoder. The process starts again at the root with the next bit.

**Scenario1:**

**16x50(Threshold=0.1), M=16 and N=50**

Entropy is: 0.96, Average length is: 12.50
Time taken for compression = 62.882113 seconds
compression ratio= 3.028064
Time taken for Decompression = 84.621934 seconds
PSNR= 29.028874, SNR = 23.60213. Took 89.094233 seconds to run.
Figure 57 (a) illustrates 16x50, if threshold=0.1, figure 57(b) illustrates Quadtree Decomposition, and figure 57(c) illustrates Decompressed image of 16x50, if threshold=0.1.

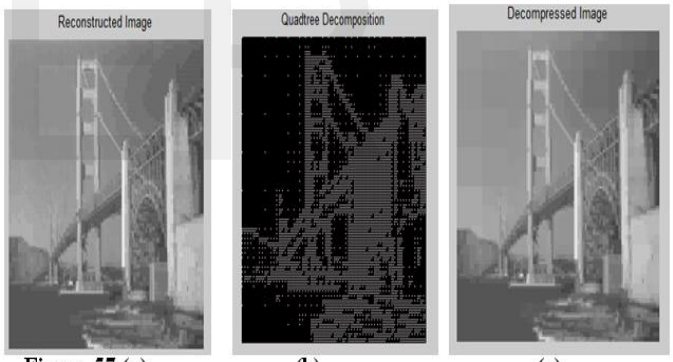


Figure 57 (a) (b) (c)

**16x50 (Default Threshold=0.5)**

Entropy is: 3.31, Average length is: 0.25
Time taken for compression = 43.151830 seconds
compression ratio= 69.552666
Time taken for Decompression = 2.179706 seconds
PSNR= 22.724171, SNR = 17.29743, Took 6.924208 seconds to run
Figure 58 (a) illustrates 16x50, if threshold=0.5, figure 58 (b) illustrates Quadtree Decomposition, and figure 58(c) illustrates Decompressed image of 16x50, if threshold=0.5.

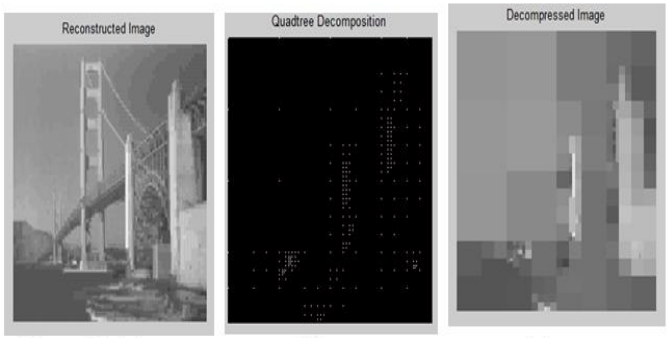


Figure 58 (a) (b) (c)

**Scenario2:**
**1024x25(Threshold=0.1)**
Entropy is: 0.96, Average length is: 12.50
Time taken for compression = 17.358541 seconds
compression ratio= 2.978418
Time taken for Decompression = 98.211322 seconds
PSNR= 25.126430, SNR = 19.69969 Took 103.814852 seconds to run
Figure 59(a) illustrates 1024x25, if threshold=0.1, figure 59(b) illustrates Quadtree Decomposition, and figure 59(c) illustrates Decompressed image of 1024x25, if threshold=0.1.

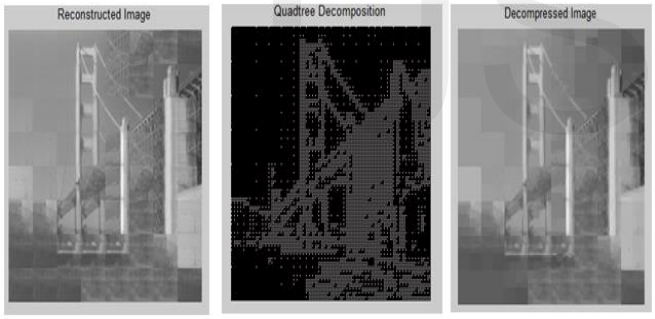


Figure 59 (a) (b) (c)

**1024x25(Default Threshold=0.5)**
Entropy is: 3.41, Average length is: 0.50
Time taken for compression = 229.637541 seconds
compression ratio= 69.331923
Time taken for Decompression = 2.123601 seconds
PSNR= 22.233890, SNR = 16.80715. Took 6.866262 seconds to run.
Figure 60(a) illustrates 1024x25, if threshold=0.5, figure 60(b) illustrates Quadtree Decomposition, and figure 60(c) illustrates Decompressed image of 1024x25, if threshold=0.5.

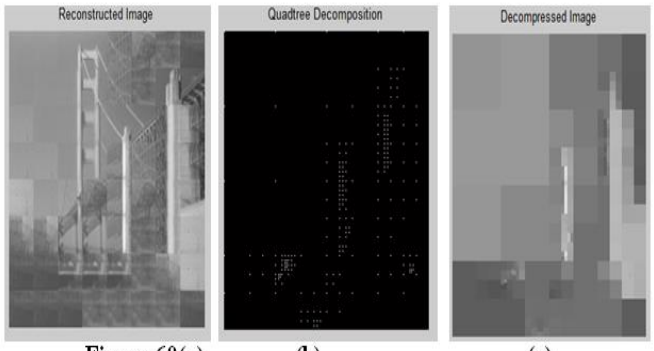


Figure 60(a) (b) (c)

As, we see in this section that scenario 1(figure 57(c)) showing better image quality result.

## 9. FUTURE SCOPE

Tremendous scope of research is going in the field of image compression. In future more parameters may be added to extend the analysis of Image compression for the images.

## 10. CONCLUSION

The image compression helps to reduce the size of the image, so that the compressed image could be sent over the computer network from one place to another in short amount of time. Also, the compressed image helps to store more number of images on the storage device [13][14].

In this research paper, the dimensional vectors or blocks for a codebook size of 25 and 50 is applied in eight different scenarios for lossless Huffman coding using K-Means algorithm in spatial and frequency domain.

In each scenario, analyzed and observed that when the block size is bigger, then the increase in quality for the reconstructed image results in a smaller SNR. Thus, Compression Ratio increases and the value of the Bit Rate decreases. Therefore, the quality of the image is not better.

In one of these scenarios, analyzed and observed that when size of block is smaller, the good quality for the reconstructed image results in a higher SNR for that reconstructed image. However, the compression ratio decreases and the value of the Bit Rate increase. Therefore, the quality of the image becomes better. Consequently, a better quality for the reconstructed image results in higher SNR results. This case also results in a lower compression ratio and a higher bit rate.

Finally, scenario 8, image has the best performance from all other scenarios. In fact, scenario 8 when M=16, and N=50 from figure 3, image has a higher SNR, which shows a better quality of reconstructed image for loss Huffman coding using K-Means algorithm in spatial and frequency domain. Thus, Compression Ratio decreases and the value of the Bit Rate increases. It shows more MSE and less PSNR. Other observations learned from obtained results that smaller vector dimensions leads to higher

image quality like in scenario 8. However, when the dimension of vector is small, then the complexity of encoding of Vector Quantization increases. The main thing, observed that there is a tradeoff between the performance of Vector quantization compression and the time spent in looking up the codebook. Comparing scenario 8 with other scenarios that larger vector dimensions leads to lower image quality.

Then, also computed the Histogram equalization (HE) of the original from figure 2 and reconstructed image(16x50 of best reconstructed image quality) from figure 3 for adjusting image intensities to enhance contrast.

Therefore, also applied the discrete fourier transform and fourier spectrum in spatial filtering and frequency filtering domain. In frequency filtering domain, used three types of low pass and high pass filters such as Ideal, Gaussian, and Butterworth for the image to check the results which type of low pass or high pass filter is the best one. In fact, in this research paper that high pass filter give emphasis to the higher frequencies in the image in which the high pass image is then added to the original image to obtain a sharper image. It may be interesting to research with width and frequency threshold of the Butterworth or the Gaussian high pass filters.

Finally, computed all the performance metrics with the execution time such as Bit Rate, Compression Ratio, SNR, PSNR, and MSE to see which one of the scenario is the best for image quality using dimensional vectors or blocks for a codebook size of 25 and 50. In this research paper, the Huffman coding analysis from provided results with the help of MATLAB implementation using K-Means technique in spatial and frequency domain based on performance metrics that more the compression ratio of Huffman coding , the lesser will be the entropy and higher average length as by if setting the threshold value to be 0.1. Therefore, PSNR and SNR become increases to show better image quality as we can see it in figure 57(c), and hence less Compression ratio. Furthermore, lesser the entropy, so better will be the image compression using K-Mean algorithm technique for lossless Huffman coding in spatial and frequency domain.

## ACKNOWLEDGEMENT

This research paper is to dedicate my acknowledgment to my advisor Dr. Jung H. Kim. I would like to thank Dr. Jung H. Kim for his most support and encouragement in understanding low-pass and high pass filters in frequency domain.

## References

[1]Raman Maini and Himanshu Aggarwal, "A Comprehensive Review of Image Enhancement Techniques", Journal of Computing, Vol. 2, Issue 3, March 2010, ISSN 2151-9617.

[2] Komal Vij, Yaduvir Singh, "Comparison between Different Techniques of Image Enhancement" International Journal of VLSI and Signal Processing Applications, Vol. 1, Issue 2 , May 2011,(112-117) ISSN 2231-3133.

[3] Rajesh Garg, Bhawna Mittal, Sheetal Garg, "Histogram Equalization Techniques for Image Enhancement", IJECT Vol. 2, Issue 1, March2011, ISSN 2230-9543.

[4] W. R. Hamilton 'Elements of Quaternions' London, U.K.: Longmans Green, 1866.

[5] Arun R, Madhu S. Nair, R.Vrinthavani and Rao Tatavarti."An Alpha Rooting Based Hybrid Technique for Image Enhancement".Online publication in IAENG, 24th August2011.

[6] S. J. Sangwine and T. A. Ell, " The discrete Fourier transform of a colour image", in Proc. Image Processing II Mathematical Methods, Algorithms and Applications, J.M. Blackedge and M.J.Turner,Eds.,Chichester, U.K., 2000, pp.430-441.

[7] Y.-T. Kim, "Contrast enhancement using brightness preserving bi histogram equalization," IEEE Trans. on Consumer Electronics, vol. 43,no. 1, pp. 1-8, Feb. 1997

[8] Y. Wang, Q. Chen, and B. Zhang, "Image enhancement based on equal area dualistic sub-image histogram equalization method," IEEE Trans. on Consumer Electronics, vol. 45, no. 1, pp. 68-75, Feb. 1999.

[9] S. J. Sangwine, "Fourier transforms of colour images using quaternion, or hypercomplex, numbers," Electron. Lett. vol. 32, no. 21, pp.1979–1980, Oct. 1996.

[10] Abdullah-Al-Wadud, M., Kabir, Md. Hasanul., Dewan, M. Ali Akber. and Chae, Oksam. (2007) "A Dynamic Histogram Equalization for Image Contrast Enhancement" IEEE 2007.

[11] Ell T.A., "Quaternion-Fourier transforms for analysis of two-dimensional linear time-invariant partial differential systems," in Proc. 32nd Con. Decision Contr., Dec. 1993, pp. 1830-1841.

[12] A. M. Eskicioglu, and P. S. Fisher, "Image quality measures and their performance," IEEE Trans. Commun., vol. 43, no. 12, pp. 2959-2965, Dec. 1995.

[13] Gonzalez, R.C. and Woods, R.E., Digital Image Processing 2nd ed., Pearson Education, India, 2005.

[14] Singara Singh , R. K. Sharma, M.K. Sharma, Use of Wavelet Transform Extension for Graphics Image Compression using JPEG2000 Framework, International Journal of Image Processing, Volume 3, Issue 1, Pages 55-60, 2009.

[15] R. Gonzalez and R. Woods, Digital Image Processing, 2nd ed. Prentice Hall, Jan. 2002.

[16] Pei C., Zeng C., and Chang H., "Virtual Restoration of Ancient Chinese Paintings Using Color Contrast Enhancement and Lacuna Texture Synthesis," Computer Journal of IEEE Transactions Image Processing, vol. 13, no. 3, pp. 416-429, 2004.

[17] Pizer M., "The Medical Image Display and Analysis Group at the University of North Carolina: Reminiscences

and Philosophy," Computer Journal of IEEE Transactions on Medical Image, vol. 22, no.1, pp. 2-10, 2003.
[18] David Salomon. Data Compression : The Complete Reference. 4th edition, Springer 2007.
[19]H.B.Kekre, Tanuja K Sarode, Sanjay R Sange(2011) " Image reconstruction using Fast Inverse Halftone & Huffman coding Technique", IJCA,volume 27-No 6, pp.34-40.
[20] Manoj Aggarwal and Ajai Narayan (2000) "Efficient Huffman Decoding", IEEE Trans, pp.936-939

## BIOGRAPHIES

**Ali Tariq Bhatti** received his Associate degree in Information System Security (Highest Honors) from Rockingham Community College, NC USA, B.Sc. in Software engineering (Honors) from UET Taxila, Pakistan, M.Sc in Electrical engineering (Honors) from North Carolina A&T State University, NC USA, and currently pursuing PhD in Electrical engineering from North Carolina A&T State University. Working as a researcher in campus and working off-campus too. His area of interests and current research includes Coding Algorithm, Networking Security, Mobile Telecommunication, Biosensors, Genetic Algorithm, Swarm Algorithm, Health, Bioinformatics, Systems Biology, Control system, Power, Software development, Software Quality Assurance, Communication, and Signal Processing. For more information, contact **Ali Tariq Bhatti** at alitariq.researcher.engineer@gmail.com.

**Dr. Jung H. Kim** is a professor in Electrical & Computer engineering department from North Carolina A&T State University. His research interests includes Signal Processing, Image Analysis and Processing, Pattern Recognition, Computer Vision, Digital and Data Communications, Video Transmission and Wireless Communications.