TUTA/IOE/PCU

Journal of the Institute of Engineering
January 2019, Vol. ***15 (No. 1):*** *91-105*

Fractal Image Compression Using Canonical Huffman Coding

Shree Ram Khaitu[1], **Sanjeeb Prasad Panday**[2]

[1]Department of Computer Engineering, Khwopa Engineering College, Purbanchal University, Nepal
[2]Department of Electronics and Computer Engineering, Pulchowk Campus,
Institute of Engineering, Tribhuvan University, Kathmandu, Nepal
Corresponding Author: *sanjeeb@ioe.edu.np*

Received: *Nov 5, 2018* **Revised:** *Jan 2, 2019* **Accepted:** *Jan 4, 2019*

Abstract: Image Compression techniques have become a very important subject with the rapid growth of multimedia application. The main motivations behind the image compression are for the efficient and lossless transmission as well as for storage of digital data. Image Compression techniques are of two types; Lossless and Lossy compression techniques. Lossy compression techniques are applied for the natural images as minor loss of the data are acceptable. Entropy encoding is the lossless compression scheme that is independent with particular features of the media as it has its own unique codes and symbols. Huffman coding is an entropy coding approach for efficient transmission of data. This paper highlights the fractal image compression method based on the fractal features and searching and finding the best replacement blocks for the original image. Canonical Huffman coding which provides good fractal compression than arithmetic coding is used in this paper. The result obtained depicts that Canonical Huffman coding based fractal compression technique increases the speed of the compression and has better PNSR as well as better compression ratio than standard Huffman coding.

Keywords: Image Compression, Entropy encoding, Huffman Coding, Canonical Huffman Coding, PSNR

1. Introduction

An image is a representation of the two dimensional finite set of digital values which are termed as pixels (picture element). The computer images are digitized which converts real world color image to numeric computer data consisting of rows and columns of color samples measured from the original image. A natural image consists of the sub sections where the picture elements of subsections have great self-similarity with each other termed as Partitioned Iterated Function System. The conversion parameters of the image can be used to be effective in compressing image which results the redundancy of images can be replaced and increased by some conversion [6, 9, 10].

Fractal image compression is one of the significant compression techniques as it has better rate of compression than that of JPEG but fractal image compression consumes more time on searching conversion replacement. Genetic algorithm is a global search method that imitates the natural genetic process which can be applied for natural images as it has multiple numbers of features. The main motivation of using Genetic Algorithm is that the natural image consists of number of properties which indicates that a chromosome with high fitness can be good candidate for replacing each block and best chance of getting the best replacement block with the properties like crossover, mutation. This helps to maintain population diversity [7, 9, 10]. Huffman coding techniques are better for fractal image compression but the process of decompression is simply a matter of translating the stream of prefix codes to individual bytes values i.e. before translating the stream of prefix codes to individual byte values, the Huffman tree must be somehow reconstructed and the overhead in information reconstruction could amounts to several kilobytes. Therefore, the main objective of this paper is to reduce the overhead in information reconstruction with the use of Canonical Huffman Coding which may precisely reconstruct with bits of information and allocates the less amount of memory than that of Huffman coding [9].

2. Methodology

2.1 Compression

Image compression is minimizing the size in bytes of a graphics file without degrading the quality of the image to an unacceptable level. This paper is based on the Huffman coding where the Huffman codes are generated then the codes are coded using Canonical Huffman Coding which reduce the size of data required for decompression and it can be visualized in block diagram as prescribed in figure.

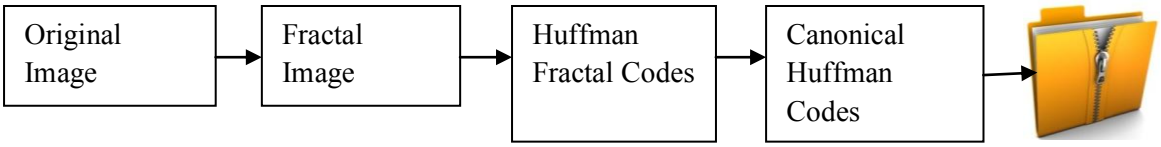

Fig. 1: Block Diagram of Compression Process

2.1.1 Original Image

The standard means of organizing and storing digital images is termed as Image File Format where the image files are composed of digital data which can be rasterized for use on a computer display or printer. It may store data in uncompressed, compressed, or vector formats [1, 11]. A compression technique stores either an exact representation or an approximation of the original image in a smaller number of bytes that can be expanded back to its uncompressed form with a corresponding decompression technique. Considering exactly the same compression, number of pixels, and color depth for two images, different graphical complexity of the original images may also result in very different file sizes after compression due to the nature of compression

techniques. Vector images, unlike raster images, can be any dimension independent of file size. File size increases only with the addition of more vectors.

2.1.2 Fractal Image Generation

Fractal image Compression is a lossy compression method for digital image, based on fractals which are best suitable for textures and natural images, relying on the fact that parts of an image often resemble other parts of the same image. Fractal methods convert these parts into mathematical data called "fractal codes" which are called as fractal image in two dimensional form and are generated by using various kinds of transform in an original image like DCT, DWT, FFT etc. and in paper Discrete Cosine Transform has been used to generate fractal image . The discrete cosine transform (DCT) helps separate the image into parts (or spectral sub-bands) i.e. fractals of differing importance (with respect to the image's visual quality). The DCT is similar to the discrete Fourier transform. [3, 8, 9]. It transforms a image from the spatial domain to the frequency domain which are used to recreate the encoded image. The general equation for a 1D (*N* data items) DCT is defined by the equation.

$$F(u) = \left(\frac{2}{N}\right)^{\frac{1}{2}} \sum_{i=0}^{N-1} A(i).\cos\left[\frac{\pi.u}{2N}(2i+1)\right] f(i) \quad (1)$$

where

$$A(i) = \begin{Bmatrix} \frac{1}{\sqrt{2}} & \text{for } \delta = 0 \\ 1 & \text{otherwise} \end{Bmatrix} \quad (2)$$

The general equation for a 2D (*N* by *M* image) DCT is defined by the equation (2)

$$F(u,v) = \left(\frac{2}{N}\right)^{\frac{1}{2}} \left(\frac{2}{M}\right)^{\frac{1}{2}} \sum_{i=0}^{N-1} \sum_{j=0}^{M-1} A(i)\,.\,A(j).\cos\left[\frac{\pi.u}{2N}(2i+1)\right] \cos\left[\frac{\pi.v}{2M}(2i+1)\right] f(i,j) \quad (3)$$

2.1.3 Huffman Fractal Codes

Huffman Fractal codes are generated using Huffman encoding techniques in the fractal image generated by DCT on the original image. Huffman coding is a clever method to construct a dictionary that is in some sense optimal for the data at hand. The method takes as input an alphabet and the probabilities with which each letter might occur in the data. The higher the probability, the shorter the code-sequence for this letter will be. Let us imagine that an image is to be encoded using Huffman Coding technique and its ASCII sequence to be stored in an memory is expressed as the sequence of number as "13371545155135706347". If stored as an ASCII sequence, the string would take 20 Bytes of storage. Noting that we have only eight "letters" in our "alphabet", i.e. {0, 1, 2, 3, 4, 5, 6, 7}, a more compact way to store the string would be to use 3 Bit sequences for each letter. The binary encoding of the original message would be: 001 011 011 111 001 101 100 101 001 101 101 001 011 101 111 000 110 011 100 111 and would take up less than 8 Bytes.

In message fives and threes occur more often than zeroes and sixes. The main idea behind entropy encoding is to use shorter codes for letters that occur often and longer codes for letters that are rare. For an instance if expressed as 7 to 0, 3 to1, 5 to 01 the decoding would not be unique. 0101 could be decoded as 735 or 573 or 55 or 7373. To make the mapping uniquely invertible we need to insist that no two codes exist in the "dictionary" where one is a prefix of the other. In the example above 0 is a prefix of 01, so the rule is violated. Dictionaries which satisfy this rule are called prefix codes or better prefix-free codes. A prefix code for our example could be which would lead to the binary encoding. Going from left to right through the binary sequence, as soon as we recognize a letter, we are sure (due to the prefix property) that it cannot be any other letter. 0 is no code 00 is 1, and no other letter starts with 00, 0 is no code, 01 is 3 and no other letter starts with 01.Altogether the encoded message is 10% shorter than the original one. Depending on the data the gain can be much bigger. But of course also the dictionary needs to be stored somewhere.

2.1.4 Canonical Huffman Codes

A canonical Huffman code is a Huffman code that allows a particularly compact way of storing the information needed to decode it because the codes are lexically ordered by length. The name canonical neither comes from the copy company nor from church, here and in mathematics canonical denotes one of many alternatives that can be distinguished since it follows simple rules. It should also be mentioned that the code lengths are the same as with Huffman codes since these are canonical Huffman codes. So there is no loss in compression when using these codes.

2.2 Decompression

After the compression of the image we are to move on the decompression of the compressed image which may be saved in the memory or are transferred from the means of communication. So, the decompression is also a necessary part and the method to decompress the compressed image is shown in Fig 2 which includes series of process and is vital for proper construction of the required image. The decompression process seems to be some how easy if there is only use of the Huffman codes but as the codes may be so much to the similar of the Huffman codes there occurs the dilemma to choose the efficient codes so genetic algorithm is being applied to choose best code blocks. The decompression process consists of series of steps which are expressed in sequential order.

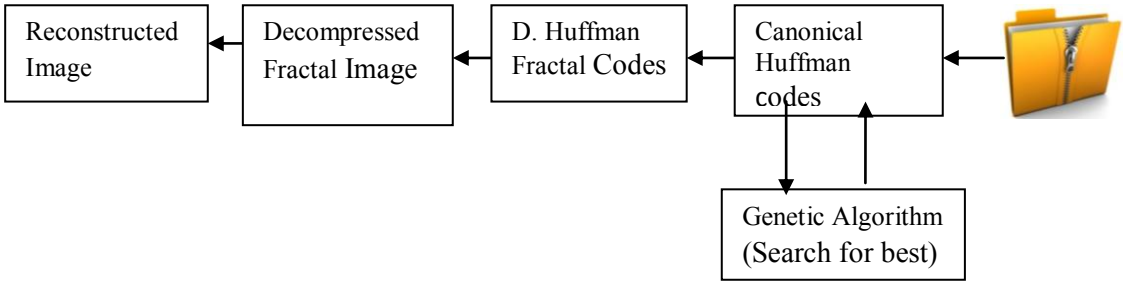

Fig. 2: Block Diagram for Decompression Process

2.2.1 Compressed File

After the completion of the compression of the given image we get a compressed file which is different than that of the normal compressed format like .rar, .zip, .tar etc. In this research there is use of the compressed file in the form of the text file where the symbols that are generated, are placed in symbol text file and the bit stream of the corresponding symbols are placed in the Huffman code text file(this is used to compare the Canonical Huffman result as Huffman is the standard method) and the Canonical Huffman code text file which do not contains the bit stream as like that of the Huffman Codes, instead of that it contains the bit stream length of the corresponding symbol. For instant if Symbol 'X' has bit stream of Huffman code as "1100101" then the Canonical Huffman code uses only the length of bit stream for compression i.e. "7" (length of "1100101").

2.2.2 Canonical Huffman Codes

Canonical Huffman codes are only the length of the bit stream which are corresponding to the symbol that are used during the compression of the image and the Huffman tree are not required that are required for reconstruction like in Huffman coding. So for generating the Huffman code form the Canonical Huffman codes a simple method is used which generate the codes which are slightly different from each other in canonical order i.e. the use of Gray Code which generates the code which are slightly different with each other.

2.2.3 Genetic Algorithm

GA is a search and optimization method developed by mimicking the evolutionary principles and chromosomal processing in natural genetics. GAs are general purpose optimization techniques based on principles inspired from the biological evolution using metaphors of mechanisms such as natural selection, genetic recombination and survival of fittest. They are member of a wider population of algorithm, Evolutionary Algorithm (EA). The idea of evolutionary computing was introduced in the year 1960 by I. Rechenberg in his work “evolution strategies” (“Evolutions strategies” in original). His idea was then developed by other researchers. Genetic Algorithm (GA) was invented by John Holland [6] and thereafter numbers of his students and other researchers have contributed in developing this field. With the advent of the GA, many non-linear, large-scale combinatorial optimization problems in power systems have been resolved using the genetic computing scheme. The GA is a stochastic search or optimization procedure based on the mechanics of natural selection and natural genetics. The GA requires only a binary representation of the decision variables to perform the genetic operations, i.e., selection; crossover and mutation. Especially GA is efficient to solve nonlinear multiple-extreme problems and is usually applied to optimize controlled parameters and constrained functions.

2.2.4 Reconstruction of Image

When the fractal image is generated then it is encoded with Canonical Huffman Encoding technique. Then it is transferred if necessary to get original image from the encoded image the inverse of the encoding is performed and that is called decoding. So decoding is to be performed

to generated original image. The decoding is a bit trickier. The Fractal image which are obtained after the decoding of the encoded image is to be converted into the image we required and for this purpose the inverse of the fractal image generation is performed to reconstruct the image, i.e. using the Inverse Discrete Cosine Transform, i.e. *inverse* 1D DCT transform is simple $F^{-1}(u)$ and *inverse* 2D DCT transform is simple $F^{-1}(u, v)$.

$$F(x,y) = \frac{2}{\sqrt{MN}} \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} C(m) \,.\, C(n) \,.\, F(m,n) \,.\, C \tag{4}$$

where $$C = cos\left[\frac{\pi.m}{2M}(2x+1)\right] cos\left[\frac{\pi.n}{2M}(2y+1)\right] \tag{5}$$

3. Results

Eleven images has been used which includes seven color images namely Lena, Onion, Peppers, Cup, Shreeram, Anime, and Redeyes. Some of them are expressed in figure and the remaining four images are gray scale image namely Mandi, Cameraman, Pout, Tires.

Fig. 3.1: Lena

Fig. 3.2: Cup

Fig. 3.3: Mandi

The test images consist of four gray scale image i.e. two dimensional image and 7 color image i.e. three dimensional image so for performing the DCT and to encode the image it is to be converted into two dimensional images. For this process the color image are to be broken into three planes i.e. into Red plane (R), Green plane (G) and Blue plane (B) if it is RGB image or into Luma Component Plane(Y), Blue Difference plane (Cb) and Red Difference plane (Cr) if it is YCbCr image. Some of the image are expressed as

Fig. 3.4: First Plane, Second Plane and Third Plane of Lenna

Fig. 3.5: First Plane, Second Plane and Third Plane of Cup

The test images are to be converted into fractal images so as to compress so DCT is applied in the images and fractal images are generated. For a color image it is to be applied three times i.e. DCT is to be applied in the three planes of the color images and the output of the images after DCT is applied are expressed which represents the DCT in each plane of Colored image, then they are combined which represents the combination of the each planes to form DCT of Color image.

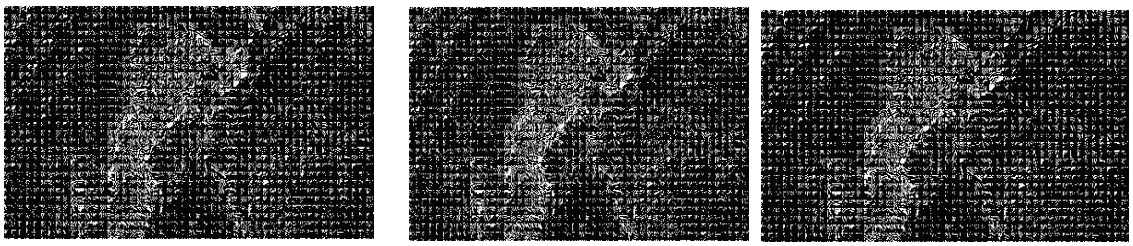

Fig. 3.6: DCT on First Plane, Second Plane and Third Plane of Lenna

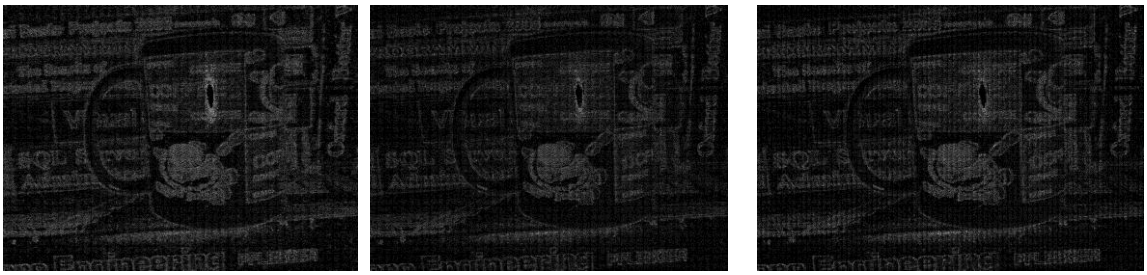

Fig. 3.7: DCT on First Plane, Second Plane and Third Plane of Cup

Fig. 3.8: DCT on Lenna

Fig. 3.9: DCT on Cup

Fig. 3.10: DCT on Mandi

After the generation of the fractal image the fractal images are to be compressed and they are compressed using the standard Huffman algorithm and the Canonical Huffman Coding. The Compression results are prescribed in Table 3.1. The Table 3.1 shows that the compression ration of the Canonical Huffman Coding is better than that of the Normal Huffman Coding but it also specifies that the originality of the image is mostly saved in the compression method of the Standard Huffman coding. So the reconstruction of the images from the compressed file there will be sure more loss of data in case of Canonical Huffman Coding but as this research is based on the compression so it has not focused more in case of reconstruction. The reconstructed images from the compressed files after application of IDCT can be expressed which represents the reconstruction of three planes of the color image, then combining three planes which

represents the combine reconstruction of color image. And for the color image reconstruction the left image represents the image reconstruction using Huffman coding and right represents reconstruction using Canonical Huffman coding.

Table 3.1 Compression ratio of Compression

S. N.	File Name	File Size	Compressed Size			Compression Ratio	
			Symbol	H. Codes	C. Codes	Huffman	C.Huffman
1	Lenna.jpg	31.7 KB	21 KB	9 KB	2 KB	0.9463	0.7255
2	Peppers.png	280 KB	230 KB	42 KB	6.5 KB	0.9714	0.8446
3	Onion.png	43.5 KB	29 KB	11 KB	3.5 KB	0.9195	0.7471
4	Cup.jpg	655 KB	560.5 KB	72.8 KB	8.2 KB	0.9668	0.86.82
5	Shreeram.jpg	25.5 KB	18 KB	7 KB	2.2 KB	0.9803	0.7921
6	Anime.jpg	383 KB	312 KB	69 KB	7.1 KB	0.9947	0.8331
7	Redeyes.jpg	90.1 KB	74.8 KB	12.5 KB	1.1 KB	0.9689	0.8423
8	Mandi.tif	81 KB	67 KB	11.3 KB	1.5 KB	0.9666	0.8456
9	Cameraman.tif	63.7 KB	49.6 KB	10. 7 KB	2.3KB	0.9466	0.8147
10	Pout.tif	67.3 KB	56 KB	9.8 KB	1.3 KB	0.9777	0.8514
11	Tire.tif	46.5 KB	34 KB	9.4 KB	2.1 KB	0.9333	0.7763

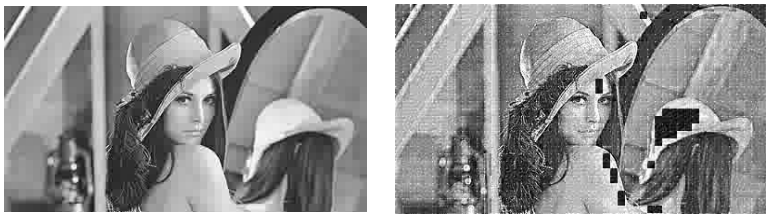

Fig. 3.11: Reconstruction of image for First Plane of Lenna

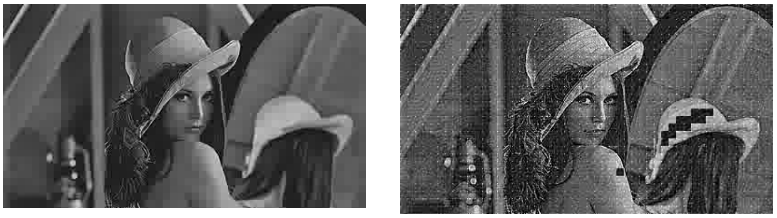

Fig. 3.12: Reconstruction of image for Second Plane of Lenna

Fig 3.13 Reconstruction of image for Third Plane of Lenna

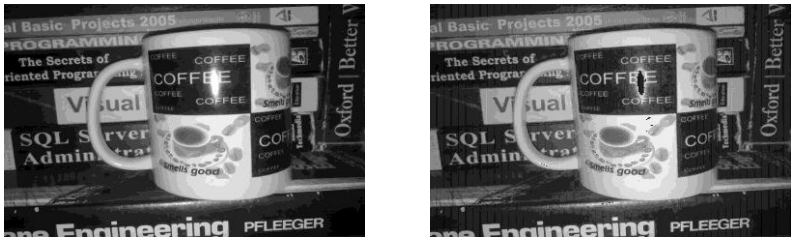

Fig 3.14 Reconstruction of image for First Plane of Cup

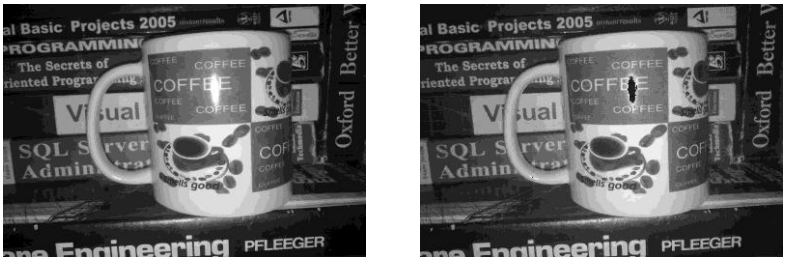

Fig 3.15 Reconstruction of image for Second Plane of Cup

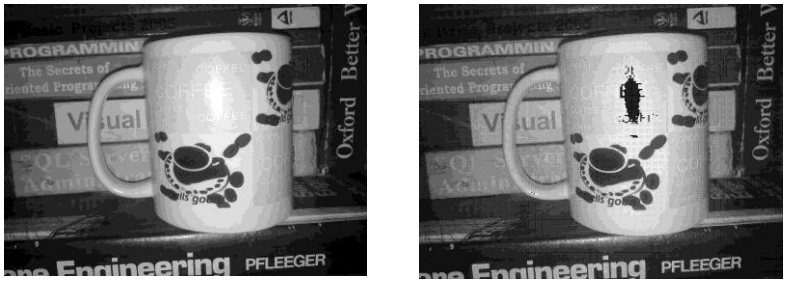

Fig 3.16 Reconstruction of image for Third Plane of Cup

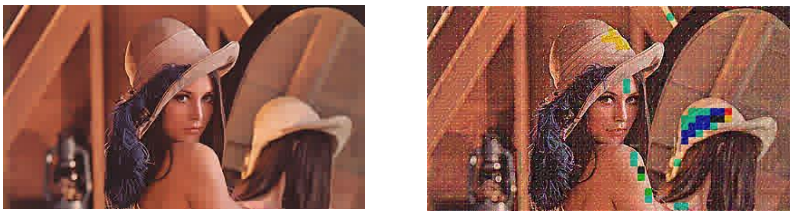

Fig 3.17 Combined Reconstruction of image lenna

Fig 3.18 Combined Reconstruction of image Cup

Fig 3.19: Reconstruction of image Mandi

From the above result of the reconstruction of the image from the compression of the images using Huffman Coding and Canonical Huffman coding it clearly states that Canonical Huffman coding is better of compression ration and better for those image whose brightness is equally distributed which can be clearly seen in the reconstruction of image Cup.jpg and Mandi.tif. For the lossless compression the PSNR of the any Compression is assumed as below 30 and for that of the lossy Compression it is considered as above 30. Peak signal-to-noise ratio, often abbreviated 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 representation. Because many signals have a very wide dynamic range, PSNR is usually expressed in terms of the logarithmic decibel scale. PSNR is most commonly used to measure the quality of reconstruction of lossy compression codec's (e.g., for image compression). The signal in this case is the original data, and the noise is the error introduced by compression. When comparing compression codec's, PSNR is an approximation to human perception of reconstruction quality.

Although a higher PSNR generally indicates that the reconstruction is of higher quality, in some cases it may not. One has to be extremely careful with the range of validity of this metric; it is only conclusively valid when it is used to compare results from the same codec (or codec type) and same content. PSNR is most easily defined via the mean squared error (MSE) and mathematically expressed in equation (7). The mathematical expression for PSNR is given in the equation (6).

$$\mathrm{PSNR} = 10\log_{10}\left(\frac{\mathrm{MAX_i}^2}{\mathrm{MSE}}\right) \tag{6}$$

where $$\mathrm{MSE} = \frac{1}{\mathrm{M}*\mathrm{N}\sum_{i=1}^{M}\sum_{j=1}^{N}[x(i,j)-x'(i,j)]^2} \tag{7}$$

And x is original image and x' is reconstructed image. Here MAX_i is the maximum possible pixel value of the image. When the pixels are represented using 8 bits per sample, this is 255. More generally, when samples are represented using linear PCM with B bits per sample, $\mathrm{MAX_i}$ is 2^B-1. For color images with three RGB values per pixel, the definition of PSNR is the same except the MSE is the sum over all squared value differences divided by image size and by three. Alternately, for color images the image is converted to a different color space and PSNR is reported against each channel of that color space, *e.g.,* YCbCr or HSL. Typical values for the PSNR in lossy image and video compression are between 30 and 50 dB, provided the bit depth is 8 bits, where higher is better. For 16-bit data typical values for the PSNR are between 60 and

80 dB. Acceptable values for wireless transmission quality loss are considered to be about 20 dB to 25 dB. In the absence of noise, the two images I and K are identical, and thus the MSE is zero. In this case, the PSNR is infinite or undefined, *i.e.* Division by zero). And the PNSR for the above images after its reconstruction is expressed in table 3.2.

Table 3.2 PSNR between Compressed and Original image

S.N.	File Name	PNSR (Huffman) (dB)	PNSR (C.Huffman) (dB)	Deivation
1	Lenna.jpg	13.3805	13.3798	0.0007
2	Peppers.png	10.5088	10.5052	0.0036
3	Onion.png	18.2056	18.2025	0.0031
4	Cup.jpg	-4.3651	-4.3637	-0.0014
5	Shreeram.jpg	8.6425	8.6360	0.0065
6	Anime.jpg	0.9204	0.9177	0.0007
7	Redeyes.jpg	8.8711	8.8615	0.0096
8	Mandi.tif	1.4395	1.4400	-0.0005
9	Cameraman.tif	17.5892	17.5885	0.0007
10	Pout.tif	17.7645	17.7659	-0.0014
11	Tire.tif	23.3079	23.3069	0.0010

The above table indicates the PSNR of the algorithm in compare with the Standard Huffman Coding and the is the difference in value i.e. deviation is positive in all cases except that of the PSNR for Cup.jpg, Mandi.tif and Pout.tif which clarifies that for the image whose brightness is equally distributed Canonical Huffman coding is better than that of Standard Huffman Coding. The Structural Similarity (SSIM) index is a method for measuring the similarity between two images. The SSIM index can be viewed as a quality measure of one of the images being compared, provided the other image is regarded as of perfect quality. The SSIM metric is based on the evaluation of three different measures, the luminance, contrast, and structure comparison measures which are computed as:

$$l(x,y) = \frac{2\mu_x\mu_y + C_1}{\mu^2{}_x\mu^2{}_y + C_1}, \tag{8}$$

$$c(x,y) = \frac{2\sigma_x\sigma_y + C_2}{\sigma^2{}_x\sigma^2{}_y + C_2}, \tag{9}$$

$$s(x,y) = \frac{\sigma_{xy} + C_3}{\sigma_x\sigma_y + C_3}, \tag{10}$$

where $\mathbf{x}$ and $\mathbf{y}$ correspond to two different signals that we would like to match, i.e. two different blocks in two separate images, μ_x , $\sigma^2{}_x$, and σ_{xy} the mean of $\mathbf{x}$, the variance of $\mathbf{x}$, and the covariance of $\mathbf{x}$ and $\mathbf{y}$ respectively, while C_1, C_2, and C_3 are constants given by

$$C_1 = (K_1 L)^2 \tag{11}$$

$$C_2 = (K_2 L)^2 \tag{12}$$

$$C_3 = C_2/2 \tag{13}$$

L is the dynamic range for the sample data, i.e. $L = 255$ for 8 bit content and $K_1 << 1$ and $K_2 << 1$ are two scalar constants. Given the above measures the structural similarity can be computed as

$$\mathrm{SSIM}(x,y) = [l(x,y)]^{\propto}.[c(x,y)]^{\beta}.[s(x,y)]^{\gamma} \tag{14}$$

where α, β, and γ define the different importance given to each measure. The MS-SSIM metric, on the other hand, is an extension of the SSIM which computes these measures at various scales and combines them using an equation of the form:

$$\mathrm{MSSSIM}(x,y) = [l_M(x,y)]^{\propto_M}.\prod_{j=1}^{M}[c_j(x,y)]^{\beta_j}.[s_j(x,y)]^{\gamma_j} \tag{15}$$

where M corresponds to the maximum scale we have used for our computation, while j=1 corresponds the original resolution of the image. In general, it is considered that these metrics perform as well as or better compared to PSNR and are used in several applications for video quality evaluation purposes. The SSIM between the reconstructed images can be calculated and expressed in table 3.3.

Table 3.3 SSIM between Compressed and Original image

S.N.	File Name	SSIM (Huffman)	SSIM (C.Huffman)	Deivation
1	Lenna.jpg	0.770179	0.524919	0.245260
2	Peppers.png	0.878744	0.198670	0.680074
3	Onion.png	0.826239	0.454548	0.371691
4	Cup.jpg	0.879213	0.152794	0.726419
5	Shreeram.jpg	0.763804	0.350797	0.413007
6	Anime.jpg	0.838720	0.239147	0.599573
7	Redeyes.jpg	0.968745	0.276311	0.692434
8	Mandi.tif	0.771977	0.069928	0.702049
9	Cameraman.tif	0.819090	0.530060	0.289030
10	Pout.tif	0.855995	0.597100	0.258895
11	Tire.tif	0.791400	0.285017	0.506383

The above table indicates the SSIM of the algorithm in compare with the Standard Huffman Coding and the is the difference in value i.e. deviation is positive in all cases but deviation for Cup.jpg, Mandi.tif is very high in compare to others which clarifies that the comparison for the PSNR is also similar i.e. for the image whose brightness is equally distributed Canonical Huffman coding is better than that of Standard Huffman Coding. DCT (Discrete Cosine Transform) is normally performed in 1D or the 2D images but the color image is the 3D in

structure which consists of three 2D planes so it was problem to perform DCT in color image. Since DCT only works on the 1D or 2D structure i.e. images so to generate the fractal image the color images are to be convert into gray scale image i.e. conversion of the 3D image into 2D image. But the problem is to covert the gray scale image into color image which is near to impossible as the gray scale image do not contain the full information of the color that is required for the color image. The example for presenting the above problem is expressed from the reconstructed image of Onion which represent original image of the onion. Some of the method to convert the gray scale image can be expressed as the use of color map function, ind2rgb, remat etc.

Fig 3.20: Original Image

Fig 3.21: Gray Scale Image

When the gray scale image is to be converted into color image we are to use the colormap function as prescribed by Matlab works but Pseudo color effect is seen the converted image which doesnot support on the reconstruction of color image.

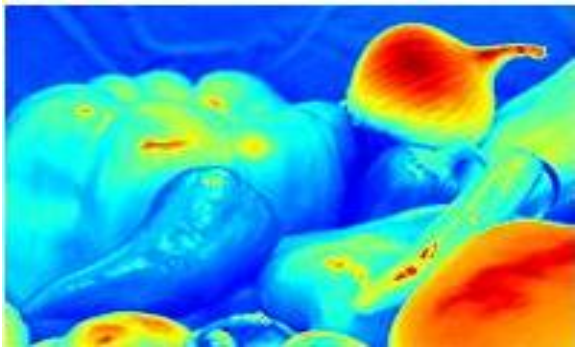

Fig 3.22: Pseudo color effect

ind2rgb is another function of the Matlab function to convert grayscale image to color but as same as colormap function this also provided same effect i.e. Pseudo color effect is seen the converted image. remat is also is another function which can be used to convert the grayscale image to color image as prescribed by Matlab works but no change is seen in converted image which can be expressed in image (Fig. 3.23).

Fig 3.23: No changes in converted image

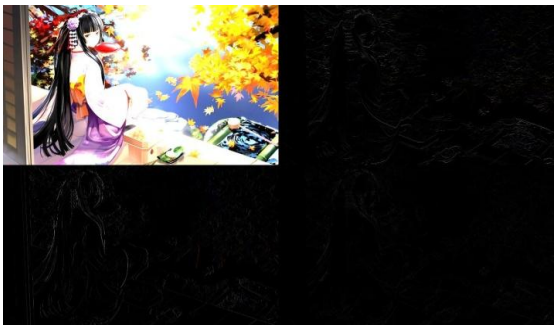

Fig 3.24: DWT of an image

So, as the process to convert the gray scale image is nearly impossible so the solution is just to apply the DCT on the Color image by dissociating the 3D color image into three different plane of 2D i.e. XYZ in to XY, XZ and YZ. And performing DCT in these three planes separately and combines it to single which is equivalent to DCT in 3D. DWT (Discrete Wavelet Transform) is generally used for the images after JPEG 2000 and DCT is used for the image before JPEG 2000. When DWT is applied in a image it provides the 4 sectioned image i.e. LL, LH, HL, HH, where most of the color information is accounted in the LL image or the section and other remaining three section consist of the gray scale image. Since the color component consist of three plane in it example RGB so it is again to be broken into 3 plane for performing Huffman coding which makes total of six images of single level decomposition of DWT which increase the complexity of algorithm. So in this research instead of the DWT DCT is used. The DWT of an image is expressed in Fig. 3.24.

4. Conclusion

In this research about eleven images are tested for compressing using Standard Huffman Coding and the Canonical Huffman Coding where the Standard Huffman Coding is the accounted for verification and validation of the Canonical Huffman Coding. Form this research we can conclude that the Canonical Huffman coding is seems to be better for the compression ratio and for compressing the images whose brightness are equally distributed and there will not be so much over head in reconstruction of image than that of Standard Huffman Coding.

This research is based on the entropy coding to encode the image using Canonical Huffman coding. This research covers the most of the aspect of the encoding as well as some extent of decoding besides this the research has some of its limitations which can be expressed as, the Canonical Huffman coding is better for compression that that of the Standard Huffman coding but Reconstructed Image is not so efficient that is given by Standard Huffman coding and Canonical Huffman Coding Depends on Standard Huffman coding for code construction and reconstruction. For efficient reconstruction of the image better optimum algorithm can be implemented. The brighter images can be utilized in the Canonical but it provides some undesirable result which can be reduced by using the Hexadecimal numbers for the generated

Huffman code in case of the length of the Huffman code which can be also easily decode can provide better PNSR, SSIM and Compression ration as well as visually appealing images.

References

[1] Annadurai S and Sundaresan M (2009), Wavelet Based Color Image Compression Using Vector Quantization and Morphology, International Conference On Advances In Computing, Communication and Control (ICAC3'09), Mumbai, Maharastra, India, 391-396.

[2] Chakrapani Y and Ranjan KS (2009), Genetic Algorithm Applied To Fractal Image Compression, ARPN Journal of Engineering and Applied Sciences **4(1):** 53-58.

[3] Grasemann U and Miikkulainen R (2005), Effective Image Compression Using Evolved Wavelets, GECCO'05, Washington, DC, USA, 1961-1968.

[4] Hassaballah M, Makky MM and Mahdy YB (2005), A Fast Fractal Image Compression Method Based Entropy, Electronic Letters on Computer Vision And Image Analysis **5(1):**30-40.

[5] Kamal NB and Priyanga P (2014), Iteration Free Fractal Compression using Genetic Algorithm for Still Color Image, ICTACT Journal on Image and Video Processing, **4(3):** 785-790**.**

[6] Omari M and Yaichi S (2012), Fractal Image Compression Using Rational Number, CTIC, Universite' d'Adrar, Alge'rie **942**: 53-57.

[7] Somasundaram K and Sumitra P (2011), RGB & Gray Scale Component on MPQ-BTC in Image Compression, International Journal on Computer Science and Engineering 3: 04-048

[8] Tong CS and Man W (2007), Adaptive Approximation Nearest Neighbor Search for Fractal Image Compression, IEEE Transactions on Image Processing **11(6)**: 605-615.

[9] Venkatasekhar D and Aruna P (2012), A Fast Fractal Image Compression Using Huffman Coding, Asian Journal of Computer Science And Information Technology **2(9):** 272-275.

[10] Wei WY (2008), An Introduction to Image Compression, Graduate Institute of Communication Engineering National Taiwan University, Taipei, Taiwan, ROC **10**: 543-548.

[11] Yerva S, Nair S and Kutty K (2011), Lossless Image Compression Based on Data Folding, CREST, KPIT Cummins Info System Ltd. Pune, India **10**: 1109