

# A Comparative Study of Lossy Image Compression using FMM & THV=10

Mr. Nitesh Agarwal<sup>1</sup>

Department of Computer Science  
Jodhpur Institute of Engineering & Technology  
Jodhpur, India

Mr. Ashutosh Vyas<sup>2</sup>

Department of Computer Science  
Jodhpur Institute of Engineering & Technology  
Jodhpur, India

Dr. Arif M. Khan<sup>3</sup>

Department of Mathematics  
Jodhpur Institute of Engineering & Technology  
Jodhpur, India

**Abstract**— A digital image is a 2D pixel matrix where each position of pixel gives a color information for image in bits format. On the basis of this bits format image is classified as 2 bit, 6 bit, 8 bit, 16 bit, 24 bit & 32 bit, but storing an image in bits format require a high amount of storage to store. As the digital devices has limited storage & transmission capability we need to compress the image by some suitable method to satisfy this limitation. Present paper embed the FMM in Lossy image compression process & analysis the output results & then embed the THV = 10 in lossy image process & finally gives a comparison between these two method on the basis of MSE, PSNR & CR & try to identify which one method is more suitable.

**Key Words:** THV, DCT, RLE, MSE, PSNR, FMM.

## 1. INTRODUCTION

Electronic snapshots taken of a scene or scanned from documents, such as photographs, manuscripts, printed texts, and artwork. The electronic snapshot is taken by image sensor device like scanner, cameras etc. These device measure the ray intensity reflected by object & store this intensity value in digits in the form of pixels. A pixel is consist of a sequence of bits, the size of sequence of bits determine the color shading in image. As the quality of image increases pixel size also increases due to increment in length of bit sequence of pixel for ex. A traditional black & white image consist 2 bit pixel i.e. only 4 shading of gray level & a true color image consist of 24 bit pixel i.e it can represents 16 millions of color but storing an image in bit formats required large amount of digital data for example a high-quality image may require 10 to 100 million bits for representation. For example, a nearly photographic image requires approximately 1,280 rows of 800 pixels each, with 24 bits of color information per pixel, that is, a total of 24,576,000 bits, or 3,072,000 bytes. The large data files associated with images thus drive the need for extremely high compression ratios to make storage (particularly of movies) practical. Digital devices & computational resources have limited communication & storage capabilities for example Without compression, a CD with a storage capacity of approximately 600 million bytes would only be able to store about 200 pictures like that above or at the 24 frames per

second rate of a motion picture it can store about 8 seconds of a movie only. Hence we need to compress image data for storing purpose as well as for communication over a network for example if there is a video conference organize by an organization which has a low bandwidth of a network, if image quality high for each frame of a video then it is difficult to deliver idea by video using low bandwidth over network provided by organization, hence organization need some image compression technique by which it compress the video before communication to provide a good communication in synchronize manner. For example when we play a YouTube video, if our communication system have enough bandwidth to play high resolution video then we can play video without any buffering problem but if bandwidth is low then YouTube use some compression method & transfer the bits of video frame in compressed format according to our network bandwidth size. By lossless image compression we get original image in decompression process without any loss of pixel value. But other images like multimedia images can be compressed using lossy image compression because the human eye is very tolerant of approximation error in an image. Hence we may decide to exploit this tolerance to produce increased compression, at the expense of image quality by reducing some pixel data or information [5].

### 1.1 Lossy Image Compression

Lossy image compression process compressed an image by adding some error in image, the error is added in such a way that entropy encoding techniques used in lossy process gives high compression ratio. There are many entropy encoding techniques available such as Run Length Encoding (RLE), Huffman Encoding, LZW (Lempel Ziv Welch) Encoding, and Area Encoding. This paper deals with RLE as a entropy encoding. The process of lossy image compression shown in fig 1. [5]

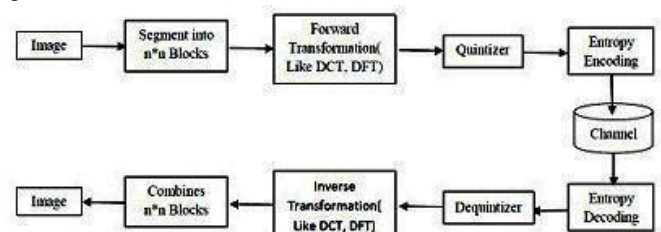


Fig 1: Lossy Image Compression

1.2 THV =10 Method

THV method traverse input matrix & take the difference  $\delta p$  between initial two nodes, & create new matrix using following condition,

If  $\delta p < 10$  then it repeat the 1<sup>st</sup> node by replacing existing nodes until  $\delta p \geq 10$ .

If  $\delta p \geq 10$  then 1<sup>st</sup> node is change with the node where  $\delta p \geq 10$ .

Using this approach this method create a good amount of repeated sequence for entropy encoding.

For example –

50	51	51	52	61	65	66	66
40	41	45	49	51	55	56	60
200	201	202	202	203	245	246	250
255	254	253	253	125	125	127	126
115	116	117	113	114	124	125	201
255	254	253	253	125	125	127	126
115	116	117	113	114	124	125	201
40	41	45	49	51	55	56	60

Table 1: Input Matrix

Table 1 shows a 2D pixel matrix but this matrix cannot be compressed using RLE because pixel value not repeated sequentially. THV method convert this matrix so that it can be compressed using RLE

50	50	50	50	61	61	61	61
40	40	40	40	51	51	51	51
200	200	200	200	200	245	245	245
255	255	255	255	125	125	125	125
115	115	115	115	115	124	124	201
255	255	255	255	125	125	125	125
115	115	115	115	115	124	124	201
40	40	40	40	51	51	51	51

Table 2: Input matrix of After THV=10 method

After the THV method pixel matrix contain a good no of repeated pixel as shown in table 2. This repeated pixel helps the RLE to compress pixel matrix. THV method used in such type of image where modifying some pixel data does not cause any big problem [1].

1.3 FMM (Five Modulus Method)

Five Modulus Method (shortly FFM) is consists of dividing the image into blocks of 8x8 pixels each. Therefore, if we can transform each number in that range into a number divisible by 5, then this will not affect the Human Visual System (HVS). Mathematically speaking, any number divided by 5 will give a remainder ranges from 0-4 (e.g., 15 mod 5 is 0, 17 mod 5 is 2, 201 mod 5 is 1, 187 mod 5 is 2 and so on). Here, we have proposed a new formula to transform any number in the range 0-255 into a number that when divided by 5 the result is always lying between 0-4. Therefore, the pixels 200,

201, and 202 are the same for the human eye. Hence, a novel algorithm have been proposed to transform each pixel in the range 0-255 into the following numbers

0,5,10,15,20,25,30,35,40,...,200, 205,210,215,...,250, 255, (i.e. multiples of 5). Actually, any number in the range 0-4 (which is the remainder of dividing 0-255 by 5) can be transformed as follows 0-> (same pixel), 1-> (-1), 2-> (-2), 3-> (+2), 4-> (+1).

FMM read each pixel value row by row & divide each pixel value by 5 & add or subtract the remainder from original pixel to get repeated pixel values. The basic idea in FMM is to check the whole pixels metrics and transform each pixel into a number divisible by 5 [6].

For Example-

50	50	50	50	60	65	65	65
40	40	45	50	50	55	55	60
200	200	200	200	205	245	245	250
255	255	255	255	125	125	125	125
115	115	115	115	115	125	125	200
255	255	255	255	125	125	125	125
115	115	115	115	115	125	125	200
40	40	45	50	50	55	55	60

Table 3: Input matrix of After FMM method

1.4 RLE (Run Length Encoding)

This is a very simple compression technique method used for compressing sequential data. Many digital image consist pixel values that are repeats sequentially for such type of image RLE is useful. In proposed THV method RLE receive sequential data from pixel matrix modified by THV method & store pixel value that repeats & no of time that pixel value repeat sequentially. For example table 2 by RLE compressed as

Pixel Value	Repetition	Pixel Value	Repetition
50	4	124	2
61	4	201	1
40	4	255	4
51	4	125	4
200	5	115	5
245	3	124	2
255	4	201	1
125	4	40	4
115	5	51	4

Table 4: Compressed Data after RLE for table 2

Table 3 required less storage space as compare to table 1. Table 1 require total 64 values to store but table 3 require only 36 values to store [5].

Compression Ratio (CR) = 64/36 = 1.78

1.5 DCT (Discrete Cosine Transform) [11]

DCT convert an image into its equivalent frequency domain by partitioning image pixel matrix into blocks of size N\*N. An image is a 2D pixel matrix hence 2D DCT is used to transform an image.

2-D DCT can be defined as

$$C(u,v) = \alpha(u)\alpha(v) \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} f(x,y) \cos \left[ \frac{\pi(2x+1)u}{2N} \right] \cos \left[ \frac{\pi(2y+1)v}{2N} \right] \quad (1)$$

for  $u, v = 0, 1, 2, \dots, N-1$ .

& inverse transformation is defined as

$$f(x,y) = \sum_{u=0}^{N-1} \sum_{v=0}^{N-1} \alpha(u)\alpha(v) c(u,v) \cos \left[ \frac{\pi(2x+1)u}{2N} \right] \cos \left[ \frac{\pi(2y+1)v}{2N} \right] \quad (2)$$

Where  $C(u, v)$  represents frequency value for  $u, v$  &  $f(x, y)$  represents pixel color value at position  $(x, y)$ .

$$\alpha(u) = \begin{cases} \sqrt{\frac{1}{N}} & \text{for } u = 0 \\ \sqrt{\frac{2}{N}} & \text{for } u \neq 0 \end{cases} \quad (3)$$

$$\alpha(v) = \begin{cases} \sqrt{\frac{1}{N}} & \text{for } v = 0 \\ \sqrt{\frac{2}{N}} & \text{for } v \neq 0 \end{cases} \quad (4)$$

1.6 Quantization

A Quantizer simply reduces the number of bits needed to store the transformed coefficients by reducing the precision of those values. Since this is a many-to-one mapping, it is a lossy process and is the main source of compression in an encoder.

The quantization matrix is designed to provide more resolution to more perceivable frequency components over less perceivable components (usually lower frequencies over high frequencies) in addition to transforming as many components to 0, which can be encoded with greatest efficiency. A DCT block is quantize using following formula

$$QDCT(i,j) = ROUND \left( \frac{DCT(i,j)}{QT(i,j)} \right) \quad (5)$$

& this QDCT block dequantize by following formula

$$DCT(i,j) = ROUND (QDCT(i,j) * QT(i,j)) \quad (6)$$

For  $i, j = 0, 1, 2, 3, \dots, N-1$

Where  $(i,t)$  define position of input & output value, QDCT is DCT block after quantization, QT is standard quantization matrices & defined as

16	11	10	16	24	40	51	61
12	12	14	19	26	58	60	55
14	13	16	24	40	57	69	56
14	17	22	29	51	87	80	62
18	22	37	56	68	109	103	77
24	35	55	64	81	104	113	92
49	64	78	87	103	121	120	101
72	92	95	98	112	100	103	99

Table 5: Standard Quantization Table [9]

1.7 THV & FMM method in Lossy Image Compression

The reason behind to use both method in lossy image compression process is to create repeated sequence, hence both method embed in lossy process, after the quantization steps as shown in fig 2, & fig 3.

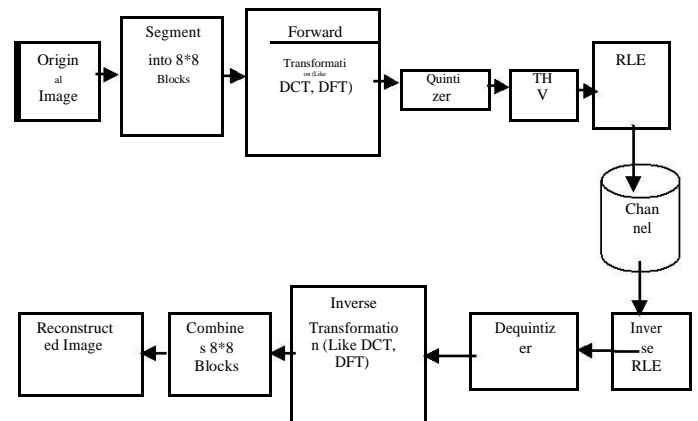


Fig 2: Lossy Image Compression with THV Method

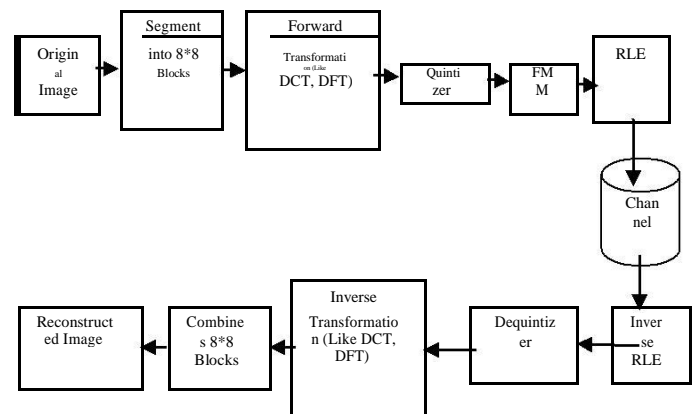


Fig 3: Lossy Image Compression with FMM Method

2. MAIN RESULTS & OUTPUTS

Table 6: THV=10 vs FMM in lossy image compression

2.1 Lossy Image compression with FMM method Steps involved in this implementation

1. Create pixel matrix of the image & divided it into blocks of size 8\*8
2. Apply FDCT (Forward Discrete Sine Transform) on each 8\*8 block of pixel matrix to get equivalent 8\*8 DCT blocks using eq. (1)
3. Apply eq. (5) on each block of DCT to get QDCT block.
4. Apply FMM on each block of QDCT to get 8\*8 FMM block.
5. Combine all FMM blocks & apply RLE on combine block & store this encoded block on secondary storage.
6. To get output image read RLE block from secondary storage & decode it to get combine FMM block & divide it into 8\*8 small transformed 8\*8 blocks.
7. To get DCT blocks apply eq. (6) on each QDCT block.
8. Apply eq. (2) on each DCT block to get IDCT blocks.
9. Combine all IDCT blocks to get pixel matrix.
10. Using pixel matrix we get required image.
11. Now we Find MSE, PSNR & CR using eq.

$$MSE = \frac{1}{H * W} \sum_{x=0}^{H-1} \sum_{y=0}^{W-1} [O(x, y) - M(x, y)]^2 \quad (7)$$






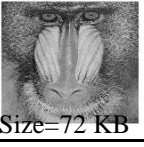

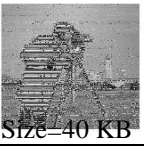







$$PSNR = 20 * \log_{10} (MAX) - 10 * \log_{10} (MSE) \quad (8)$$

$$CR = \frac{Original \ Image \ size}{Output \ Image \ size} \quad (9)$$

Where H=Height of Image, W= Width of Image, variable MAX shows max value of a pixel for example here image is 8 bit hence MAX=255.

2.2 Lossy Image compression with THV=10 method Steps involved in this implementation

1. Create pixel matrix of the image & divided it into blocks of size 8\*8
2. Apply FDCT on each 8\*8 block of pixel matrix to get equivalent 8\*8 DCT blocks using eq (1).
3. Apply eq (5) on each block of DCT to get QDCT block.
4. Apply THV algorithm on each block of QDCT to get THV block.
5. Combine each THV block & apply RLE on combine block & store this encoded block on secondary storage.
6. To get required image read encoded matrix from secondary storage & apply entropy decoding (Run Length Decoding) on that encoded matrix.
7. Divide this decoded matrix in to blocks of size 8\*8.
8. Apply eq (6) on each block to get DCT blocks.
9. Apply eq (2) on each DCT block to get IDCT blocks.
10. Combine all IDCT blocks to get pixel matrix. Using pixel matrix we get required image
11. Now we Find MSE, PSNR & CR using eq. (7), (8) & (9)

Input image Size=768 KB		Compressed image	MSE	PSNR	CR
 Lena.bmp	THV	 Size=40 KB	2037.81	15.04	19.2
	FMM	 Size=40 KB	60.69	30.30	19.2
 Baboon.bmp	THV	 Size=64 KB	3014.66	13.34	12
	FMM	 Size=72 KB	301.95	23.33	10.67
 Camera_man.bmp	THV	 Size=40 KB	5412.0	10.78	19.2
	FMM	 Size=40 KB	441.84	21.68	19.2
 House.bmp	THV	 Size=32 KB	1576.21	16.15	24
	FMM	 Size=32 KB	55.24	30.71	24
 Pappers_grey.bmp	THV	 Size=40 KB	2990.68	13.37	19.2
	FMM	 Size=40 KB	184.66	25.47	19.2

### 3. CONCLUSION

Image compression using FMM & CThV gives better compression ratio than using RLE but it adds more errors in image hence if image compression process required purely lossless image compression then both methods cannot be used with RLE. But in lossy image compression both methods can be used with RLE to get high compression ratio with less PSNR & high MSE value. FMM is a more efficient technique than CThV method because it gives almost the same compression ratio as given by CThV method but its MSE value is less & PSNR value is high w.r.t. CThV method as shown in table 6.

### REFERENCES

- [1] A. H. Hussein, S. Sh. Mahmud & R. J. Mohammed "Image Compression Using Proposed Enhanced Run Length Encoding Algorithm", *Ibn Al-Haitham Journal For Pure And Applied Science*, VOL 24(1), pp. 315-328, 2011.
- [2] A.M. Raid, W.M. Khedr, M.A. El-dosuky and Wesan Ahmed "JPEG image compression using Discrete Cosine Transform – A survey", *International Journal of Computer Science & Engineering Survey (IJCSES)*, vol 5, no 2, pp. 29-47, April 2014.
- [3] Andrew B. Watson, "Image Compression Using Discrete Cosine Transform", *NASA Ames Research Centre*, 4(1), pp. 81-88, 1994.
- [4] Anjali Kapoor and Dr. Renu Dhir, "Image Compression Using Fast 2-D DCT Technique", *International Journal on Computer Science and Engineering (IJCSSE)*, vol. 3 pp. 2415-2419, 6 June 2011.
- [5] Asha Lata & Permender Singh "Review of Image Compression Techniques", *International Journal of Emerging Technology & Advanced Engineering (IJETAEE)*, vol 3, issue 7, pp. 461-464, July 2013.
- [6] Firas A. Jassim and Hind E. Qassim, "FIVE MODULUS METHOD FOR IMAGE COMPRESSION", *signal & image processing: An International Journal (SIPIJ)*, vol 3, no.5, pp. 19-28, October 2012.
- [7] Harley R. Myler and Arthur R. Weeks "The Pocket Handbook of Image Processing Algorithms in C", ISBN 0-13-642240-3 Prentice Hall PTR Englewood Cliffs, New Jersey 07632.
- [8] Iain E.G. Richardson "H.264 and MPEG-4 Video Compression: Video Coding for Next-generation Multimedia", ISBN 0470848375, 9780470848371, Wiley, 2003.
- [9] Jesse D. Kornblum "Using JPEG quantization tables to identify imagery processed by software", *ELSEVIER, DIGITAL INVESTIGATION* 5, pp. S21-S25, 2008.
- [10] Maneesha Gupta and Dr. Amit Kumar Garg, "Analysis Of Image Compression Algorithm Using DCT" *International Journal of Engineering Research and Applications (IJERA)*, vol.2, pp. 515-521, Jan-Feb 2012.
- [11] N. Ahmed, T. Natarajan, and K.R. Rao, "Discrete Cosine Transform", *IEEE Transactions on Computers*, vol. C-32, pp. 90-93, Jan. 1974.
- [12] Nitesh Agarwal and Dr. A.M. Khan "Application of DCT in image processing", *International Journal of Engineering Research & Technology (IJERT)*, ISSN: 2278-0181, pp. 185-189, 2014.
- [13] Swati Dhamija and Priyanka Jain "Comparative Analysis for Discrete Sine Transform as a suitable method for noise estimation" *IJCSI International Journal of Computer Science Issues*, Vol. 8, Issue 5, No 3, September 2011 pp. 162-164.
- [14] V. Singh "Recent Patents on Image Compression – A Survey", *Recent Patents on signal Processing (Bentham Open)*, vol 2, pp. 47-62, 2010.