Multilevel Compression Scheme using Vector Quantization for Image Compression

S.Vimala, B.Abidha, and P.Uma

Abstract—**In this paper, we have proposed a multi level compression scheme, in which the initial codebook and the Index Map that are generated using the normal Vector Quantization are compressed in three levels. The Minimum Mean Square method that is adopted in Block Truncation Coding is used to compress the codebook in first level with some improvements. The interpolative method is used to perform second level of compression to the codebook. Finally the Search Order Coding method is used to compress the Index Map without loss of image data. The experimental results show that there is a significant improvement in bit reduction without loosing much of image data.**

Index Terms—**codebook, compression, index-map, interpolative, MMSE, SOC**

I. Introduction

Recent researches concentrate in finding ways to code images efficiently using digital techniques for image processing [1]. Generally images require large amount of data resulting in consumption of huge bandwidth and storage resources. Image compression is essential for applications such as TV transmission, video conferencing, facsimile transmission of printed material, graphics images, or transmission of remote sensing images obtained from satellites and reconnaissance aircraft [2]. Vector Quantization (VQ) [3]-[5], due to its simplicity and coding efficiency, has been successfully used in various applications involving VQ based encoding and VQ based recognition. VQ is an efficient image coding technique achieving a bit rate, less than one bit per pixel (bpp). Different types of VQ, such as classified VQ [6], [7], address VQ [8], finite state VQ [9], Side match VQ [10], Mean-removed classified VQ [11] and Predictive classified VQ [9], [11] have been used for various purposes. VQ comprises of three stages: codebook designing, image encoding and image decoding.

A Vector Quantizer Q of dimension k and size N is a mapping from a vector in k-dimensional Euclidean space points [14]. This can be expressed mathematically as $Q:R^k \rightarrow C$, where $C=\{Y_1, Y_2, Y_3, ..., Y_M\}$ and $Yi \in R^k$. The set C is called the codebook and $Yi, 1<=i<=M$ are called codewords.

The codebook generation is the important task in VQ. [12] Performance of the VQ is highly dependent on the optimality of the codebook. There are several known methods for generating a codebook [13]. LBG (Linde Buzo, Gray) [4] is the most widely referred VQ method for designing a Codebook.

The most widely used Generalized Lloyd Algorithm (GLA) starts with an initial solution, which is iteratively improved using two optimality criteria in turn until a local minimum is reached. It can also be built hierarchically. The iterative splitting algorithm [15], [16] starts with a codebook of size one, which is the centroid of the entire training set. The codebook is then iteratively enlarged by a splitting procedure until it reaches the desired size. Another hierarchical algorithm, the Pairwise Nearest Neighbor (PNN) [17], uses an opposite, bottom up approach to generate the codebook. The Vector Quantization algorithms for reducing the transmission bit rate or the storage have recently been extensively investigated for speech and image signals [18]. A fundamental goal of data compression is to reduce the bit rate for transmission or data storage while maintaining an acceptable fidelity or image quality.

In this paper, the compression rate is increased in multiple levels. The concepts of normal VQ with novel ideas incorporated, Minimum Mean Square Error (MMSE), Interpolative method and Search Order Coding (SOC) are integrated to reduce the bitrate to a greater extent. [19] The MMSE method and the Interpolative method [20] which are used in Block Truncation Coding (one other lossy image compression technique) are incorporated in VQ in the proposed scheme to reduce the bitrate to a significant extent. In the remaining part of the paper, the proposed scheme is described in Section 2, the results are discussed in Section 3 and the conclusion is given in Section 4.

II. Proposed Scheme

In the proposed scheme, VQ is performed in different levels. A novel idea called the Codebook Generation with Edge Categorization (CBEC) is introduced to generate the initial codebook. Initially, the image is normal vector quantized with the novel idea incorporated in generating the codebook. The codebook thus generated during normal vector quantization is further compressed in three levels. In the first level, the codebook is compressed using Minimum Mean Square Error (MMSE). In the second level, the codebook is compressed using Interpolation method. In the third level, the concept of Search Order Coding (SOC) is used. The codebook thus generated and compressed is optimized using K-Means clustering in order to improve the quality of the codebook. The quality of the reconstructed image depends on the size of the codebook and the quality

Manuscript received May 28, 2011; revised December 6, 2011.

The authors are with the Department of Computer Science, Mother Teresa Women's University, Kodaikanal - 624102, Tamil Nadu, India since 1999 (e-mail: vimalaharini @gmail.com, rabidha@yahoo.com, umapoomi@gmail.com).

of the codebook. The methods involved in the proposed scheme are explained in the following sections.

A. Codebook Generation with Edge Categorization (CBEC)

The image is sub-divided into 4 x 4 pixels. The block is then converted into one dimensional vector of size k (where k = 4 x 4) elements. Thus the input image is converted in to a training set of size N vectors, where N is computed using the equation (1).

$$N = (m \times m) / k \quad (1)$$

From each input block, the maximum pixel value *Max* and the minimum pixel value *Min* are calculated. If the *Min* and the *Max* values lie in the range 1 to 16, the input block belongs to the 1st cluster. If the *Min* and *Max* values lie within the range 17 to 32, the input block belongs to the 2nd cluster. Similarly 16 such clusters are formed based on the minimum and maximum values. The ranges are (1-16), (17-32), (33-48)… (225-240), (241-256). The average vectors of all the 16 clusters called the centroids are computed to form 16 code vectors of the codebook. Some of the input blocks that do not fall into any of the above clusters are used to generate the remaining (M-16) code vectors, where M is the desired codebook size. Such blocks are categorized into high detail and low detail blocks. To categorize them as high detail and low detail blocks, the mean ($\bar{x}$) for each block is computed using the equation (2). The sum S of the difference between the mean and the individual pixel value is computed using the equation (3).

$$\bar{x} = \frac{1}{m}\sum_{i=1}^{m} x_i \quad (2)$$

$$S = \sum_{i=1}^{m} abs(x_i - \bar{x}) \quad (3)$$

If S is greater than a threshold value, then the block is categorized into high detail block otherwise it is a low detail block. The threshold value depends up on the size M of the codebook. Generally the edge blocks will be of high detail blocks. The edge blocks are given preference to preserve the edges in the image. This avoids ragged edges. Hence more number of edge blocks is selected when compared to the shade blocks while filling the remaining codebook. This improves the quality of the codebook.

B. Categorization of blocks

Total number of blocks for an image of size 256 *x* 256 pixels: 4096

Lena image

No. of blocks satisfying the MinMax condition: 956
No. of edge blocks: 1520
No. of shade blocks: 1620

Cameraman image

Min-Max Blocks: 1395
Edge Block: 1512
Shade Block: 1189

The codebook thus generated is optimized using the K-Means Clustering technique. The optimization of codebook improves the quality of the reconstructed image.

C. Selection of Blocks for The Codebook

The blocks satisfying the *Minima* condition are grouped into 16 clusters and the centroids of the clusters are taken thus forming first 16 codevectors. In case of a codebook of size 256, the remaining 240 codevectors are selected from the edge and shade blocks. Of these 240, 200 vectors are taken from edge blocks arbitrarily and the remaining 40 blocks are selected from the shade blocks. Thus a codebook of desired size is generated. Similarly for codebook of other sizes, the number of different types of blocks to be selected is given in TABLE I.

TABLE I: NO. OF BLOCKS OF DIFFERENT TYPES SELECTED FOR VARIOUS CODEBOOK SIZES.

CB Size	Min-Max	Edge	Shade
64	16	40	8
128	16	100	12
256	16	200	40
512	16	284	212
1024	16	584	424

The above numbers are fixed on trial basis based on the quality of the reconstructed images.

D. K-Means Clustering

The initial codebook thus created is optimized using the K-Means clustering, called Generalized Lloyd Algorithm. The training vectors are grouped into M clusters based on the equation (4). Pick up the codebook entry $y_i(k)$. Find all the training vectors that are closer to y_i that satisfy

$$d(x_i y_{i)} < d(x_i y_{j)} \text{ for all } j \neq i \quad (4)$$

where the distance between the training vectors x_i and the code vector y_i is computed using the equation (5).

$$d(x_i, y_i) = \sum_{i=1}^{k} |x_i - y_i| \quad (5)$$

For each cluster, the centroid is computed. To compute the centroid, the Sum Vector Sum_{ij} is computed using the equation (6).

$$Sum_{ij} = \sum_{i=1}^{c_i} x_{ij} \quad where\ j = 1, 2, \ldots, k \quad (6)$$

where, c_i is the cluster strength. Now every component of the Sum Vector is divided by the cluster strength c_i to get the new *centroid of the cluster*.

$$Centroid = Sum_{ij}/c_i \quad where\ i = 1, 2, \ldots .M \quad (7)$$

The codevector y_i is replaced with the newly generated centroid to form the refined codebook. The above steps are repeated until the codebooks of the consecutive iterations converge.

Algorithm

Step1: The training vectors are grouped into M clusters based on the equations (4) and (5).
Step2: Compute the Sum Vector using the equation (6).
Step3: Compute the centroid of the each cluster using the equation (7).
Step4: Replace the existing codevectors with the new centroids to form the revised codebook.

Step5: Repeat the steps 1 through 4 till the codebooks of consecutive iterations converge.

The PSNR value of the reconstructed image using the optimized codebook is high when compared to that of the image reconstructed using the initial codebook. The size of the codebook thus generated and optimized is reduced further using the MMSE method.

E. Minimum Mean Square Error (MMSE) Method

Minimum Mean Square Error (MMSE) was proposed by Lloyd [S. P. Lloyd, 1982] for Block Truncation Coding which is one other lossy image compression technique. In 4-level MMSE technique, the *Min* and *Max* values are calculated for each block and a 4-level quantizer (a, b, c, d) is designed based on threshold value that is computed using the equation (8).

$$t_r = min + ((max- min)\ r/n) \tag{8}$$

where t_r represents the r^{th} value of the threshold and *n* is the number of quantization levels and four quantizing levels a, b, c and d are calculated using the equation (9). Quantizer values are optimized.

$$\left.\begin{aligned} a &= min \\ b &= (2\times min+max)/3 \\ c &= (max+2\times min)/3 \\ d &= max \end{aligned}\right\} \tag{9}$$

The compressed codevector is stored as a set of a bit plane and the two other statistical moments i.e. {32-bit bit plane, a, d}. While reconstructing the image, *b* and *c* can be computed from the values *a* and *d*. The entire codebook is thus a collection of sets of 2-bit bit planes and the respective statistical moments for all the codevectors. Each component of the codevector is compared against the threshold values *tr,* where r=1,2 and 3. If the component of the code vector is <= t1, then it is coded as 00. If > t1 and <= t2, then it is coded as 01. If the value is > t2 and <= t3, it is coded as 10 other wise 11. Now the coded bitplane for any codevector will be like [11, 11, 01, 00, 11, 11, 01, 00, 11, 10, 10, 00, 11, 10, 01, 00] and the codevector is replaced with the set of coded bit plane, *a* and *d*. Generally, a codeword takes 16 x 8 = 128 bits. But after compression using 4-level MMSE, the codevector requires only 32 + 8 + 8 = 48 bits. A compression rate of 62.5% is achieved.

Compression achieved as a result of Normal VQ with a codebook of size 256 codewords

Codebook Size (256 × 16 × 8) : 32768 bits
Index Map Size (64 × 64 × 8 : 32768 bits
Total Compressed Size : 65536 bits
Original image size (256 ×256 pixels) : 65536 pixels
Total number of Bits required by the original image (256 ×256 ×8) : 524288 bits
Bit-rate = Compressed Size / Original Image Size = 65536/65536 = 1.00 bpp
Compression Rate = 100-(Size of Compressed Image/Size of Original Image × 100)
= 100-((65536/524288)* 100)
= 87.50%

Compression achieved as a result of MMSE

Codebook Size (256 × 48) : 12288 bits
Index Map Size (64 × 64 × 8 : 32768 bits
Total Compressed Size : 45056 bits
Original image size (256 ×256) : 65536 pixels
Total Bits of the original image (256 ×256 × 8) : 524288 bits
Bit-rate = Total Compressed Size / Original Image Size
= 45056/65536 = 0.69 bpp
Compression Rate = 100-(Size of Compressed Image/Size of Original Image × 100)
= 100-((45056/524288)× 100)
= 91.40%

MMSE Algorithm

Step1: Get the initial codebook
Step2: For each codevector in the codebook, perform the following steps.
Step3: Calculate min and max values for each codevector
Step4: Calculate 4-level quantizer using equations (8) and (9)
Step5: Store Bitplane and quantizer levels

The codebook thus compressed using MMSE method is further compressed using the Interpolative method.

F. Interpolative Method

In this method, the bitplane generated using MMSE is further compressed by dropping the intermediate bit sequences as in Fig. 2. Hence the size of the bitplane is reduced from 32 bits to just 16 bits.

X1	**X2**	X3	**X4**
X5	X6	**X7**	X8
X9	**X10**	X11	**X12**
X13	X14	**X15**	X16

Fig 1. The pattern of dropping bits. The bold faced bits are dropped.

In decoding phase, the dropped bits are recovered by taking the arithmetic mean of the adjacent pixel values as given in equation (10).

$$\left.\begin{aligned} x_i &= \frac{1}{3}(x_{i-1}+x_{i+1}+x_{i+4}) && \text{for i=2} \\ x_i &= \frac{1}{2}(x_{i-1}+x_{i+4}) && \text{for i=4} \\ x_i &= \frac{1}{3}(x_{i-4}+x_{i+1}+x_{i+4}) && \text{for i=5} \\ x_i &= \frac{1}{4}(x_{i-4}+x_{i-1}+x_{i+1}+x_{i+4}) && \text{fori=7,10} \\ x_i &= \frac{1}{3}(x_{i-4}+x_{i-1}+x_{i+4}) && \text{for i=12} \\ x_i &= \frac{1}{2}(x_{i-4}+x_{i+1}) && \text{for i=13} \\ x_i &= \frac{1}{3}(x_{i-1}+x_{i+1}+x_{i-4}) && \text{for i=15} \end{aligned}\right\} \tag{10}$$

As a result of this method, the codebook takes only 32 (16 + 8 + 8) bits per vector.

Compression achieved as a result of Interpolative method

Codebook Size (256 ×32): 8192 bits

Index Map Size (64× 64× 8) : 32768 bits
Total Compressed Size : 40960 bits
Original image size (256 × 256) : 65536 pixels
Total Bits of the original image (256 ×256 × 8) : 524288 bits
Bit-rate = Total Compressed Size / Original Image Size
= 40960/65536 = 0.63 bpp
Compression Rate = 100-(Size of Compressed Image/Size of Original Image ×100)
= 100-((40960/524288)× 100)
= 92.19%

Further compression is done to the Index Map using the Search Order Coding (SOC). The above two levels lead to lossy compression. But compression using SOC leads to lossless compression.

G. Search Order Coding (SOC)

In SOC, the feature of inter-block correlation is exploited. There are more chances for a processed block to be similar to the neighboring blocks. The index of the current block cb is compared with the index of the neighboring blocks in the given order lb, tb, tlb, lrb as in the Fig. 2.

	ulb	***ub***	***urb***	
	lb	***cb***		

Fig. 2. Search Order of the Neighboring Indices

If the index of ***cb*** is is equal to the index of ***lb***, the index of the current block is coded as '00'. If equal to the index of block ***ub***, the index is encoded as '01'. If equal to the index of the block ***ulb***, the index is encoded as '10'. If equal to the index of the block urb, the index is encoded as '11'. If index is not equal to any of the neighboring blocks, then the original index of size 8 bits is stored (8 bits in the case of codebook of size 256). The total number of bits taken to store the encoded image is computed using the equation

$$N = N1 + N2 + N3 + N4 \quad (11)$$

where **N1**-Number of blocks whose indices are equal to the index of the left block.
N2-Number of blocks whose indices are equal to the index of the upper block
N3-Number of blocks whose indices are equal to the index of the upper left block
N4-Number of blocks whose indices are equal to the index of the upper right block

The number of blocks that are not similar to the nearest neighbors is then computed equation (12) as

$$n = (64 \times 64 - N) \quad (12)$$

With normal VQ, the bit-rate achieved with a codebook of size 256 is computed as

Codebook Size (256 ×16 × 8) : 32768 bits.
Index Map Size = 64 × 64 × 8 : 32768 bits
Total No. of bits : 65536 bits
Bit-rate (65536/(256×256)) : 1 bpp

With the proposed scheme, for the image Lena

Codebook Size (256 ×32) : 8192 bits
Index Map = (2243×8) +(1853×2) : 21650 bits
Total No. of bits : 29842
Bit-rate (29842/(256×256)) : 0.46 bpp

Compression achieved as a result of Search Order Coding

For Lena image,
Codebook Size (256 ×32) : 8192
Index Map Size ((2243×8) +(1853×2)) : 21650
Total Compressed Size : 29842
Original image size (256 ×256) : 65536
Total Bits of the original image (256 ×256 ×8) : 524288

Bit-rate = Total Compressed Size / Original Image Size = 29842/65536 = 0.46 bpp
Compression Rate = 100-(Size of Compressed Image/Size of Original Image× 100)
= 100-((29842/524288)×100)
= 94.31%

H. Decoding Stage

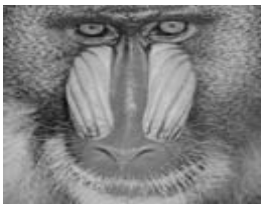
In decoding stage, to reconstruct the compressed image, the codebook is decompressed in the reverse order in which the codebook is transformed into various forms through all the levels.

Step1: The reverse of Interpolation is done in which the dropped out bit sequences are reconstructed using the equations (10).
Step2: The reverse of MMSE is performed, in which the code vector is reconstructed from the 2-bit plane and its respective statistical moments.
Step3: The Index map is reconstructed based on the similarity of the neighboring blocks.
Step4: For each index in the index map, the corresponding codevector is coded to get the approximation of the input image.

III. Results and Discussion

Experiments were carried out with standard images Cameraman, Boats, Bridge, Baboon, Lena and Kush of size 256 x 256 pixels. The input images taken for the study are given in Fig. 3.

(a) Cameraman (b) Boats (c) Bridge



(d) Baboon (e) Lena (f) Kush

Fig. 3. Input images taken for the study

The PSNR value is taken as a measure of the quality of the reconstructed image and bpp is the measure of the compression efficiency. The algorithm is implemented using MATLAB 7.0 on Windows Operating System. The hardware used is the Intel® 3.06GHz Processor with 512 MB RAM. Table 1 gives the results obtained in terms of PSNR and bpp after every level of compression for standard images with codebooks of different sizes. For a codebook of size 64, the bpp has been reduced from 0.50 to 0.26. For a codebook of size 1024, the bpp has been reduced from 2.63 to 1.04, which is a significant improvement.

In TABLE II, wit codebook of smaller sizes (64 and 128), the PSNR is not much affected, But the bpp has been reduced to half of the size that is obtained after VQEC. As the size of the codebook increases, there is a decrease in bpp as well as in PSNR.

In TABLE III, quality (PSNR) of the reconstructed images, the bpp and the Compression Rate (in %) after every level of compression are given. A maximum of 96.21% is achieved with the proposed scheme, which is an appreciable value.

Linde, Buzo and Gray (LBG), Kekre's Proportionate Error Algorithm (KPE) [22], Simple Codebook Generation (SCG) and Ordered Codebook Generation (OCG) [23], Codebook Compression (CBC) [24], Training Set Partitioning (TSP) [25] methods are some of the existing codebook generation techniques. The results obtained with them are compared with that of the proposed scheme in TABLE IV. With codebooks of lesser sizes, there is no much loss in PSNR, but the bpp has been reduced to a significant value.

In techniques such as LBG, KPE, SCG and OCG TSP, the PSNR obtained is little bit greater than that of the proposed method. But the bit-rate achieved in the proposed method is less when compared to that of all the methods. When compared to the TSP method, both the PSNR and the bpp values are better in the proposed method. With codebooks of smaller size, there is only little raise in the PSNR. But as the codebook size increases, the maximum difference in PSNR values between the existing methods and the proposed method is 3.32. But the proposed methods gives better compression rate when compared to all the existing methods mentioned in the study. In LBG method, with the codebook of size 1024, the bpp obtained is 2.63. But the bpp achieved with the proposed method is only 1.04 which is a significant improvement. A minimum of 0.27 bpp is also achieved with the proposed method with the codebook of size 64.

IV. Conclusion

In this paper, we have presented a simple and efficient coding scheme for the compression of Image. The compression is done in four levels. In the first level, normal VQ is done, where a novel idea has been incorporated in generating the initial codebook with edge categorization. In the second level, the concept of MMSE is used to compress the codebook to contain the bit plane and the statistical moments of the codevectors. In the third level, using the concept of Interpolation, further compression is done to the codebook by reducing the size of the bit plane by dropping the bit sequences in the alternate positions. Finally the concept of Search Order Coding is performed to reduce the size of the Index Map. In each level, the compression rate is improved. At the end of the compression procedure, a significant amount of compression that is up to 94.31% is achieved. After normal VQ is performed, an average compression rate of 84.6.5% is achieved. After every level of compression, the average compression rate is improved from 84.06% to 90.12%, 91.33% and 92.68% respectively. The simulation results show that the proposed scheme improves the compression efficiency by maintaining significant level of PSNR of the reconstructed image. The image crosses four levels of compression i. Normal VQ, ii. 4-level MMSE, iii. Interpolative Technique and iv. Search Order Coding. By incorporating the four levels of compression, it is noticed that a significant reduction in bit rate is achieved. The results of the proposed scheme, when compared to that of few existing methods, gives better results both in terms of bit-rate and the quality of the reconstructed images. With the higher codebook sizes, the existing methods give better PSNR values. But as far as the compression rate is concerned, only the proposed method gives a significant reduction in the bpp. This compression scheme can easily be extended to color images. This application is suitable for hand-held devices, where image processing applications play a vital role in present scenario. Since it is a lossy compression system, it is- suitable for Human Visual System (HVS), where loss in image data

does not make much difference. In this paper, we have integrated the ideas incorporated in BTC and VQ to bring out the proposed scheme. Further enhancements to this kind of techniques and when integrated with other techniques, may lead to a better scheme when compared to JPEG standard in near future.

TABLE II: QUALITY OF THE RECONSTRUCTED IMAGES AND THE BPP OBTAINED AFTER EVERY LEVEL OF COMPRESSION FOR DIFFERENT IMAGES WITH DIFFERENT CODEBOOK SIZES

CB Size	Images	CBEC		MMSE		Interpolation		SOC	
		PSNR	bpp	PSNR	bpp	PSNR	bpp	PSNR	bpp
64	Cameraman	28.43	0.50	28.16	0.42	27.46	0.41	27.46	0.23
	Baboon	34.67	0.50	34.50	0.42	34.50	0.41	34.50	0.31
	Boats	29.21	0.50	29.00	0.42	28.45	0.41	28.45	0.26
	Bridge	26.94	0.50	26.81	0.42	26.58	0.41	26.58	0.31
	Lena	31.47	0.50	31.22	0.42	31.02	0.41	31.02	0.25
	Kush	31.86	0.50	31.67	0.42	31.46	0.41	31.46	0.27
	Goldhill	34.27	0.50	34.12	0.42	33.92	0.41	33.92	0.26
	Average	**30.98**	**0.50**	**30.78**	**0.42**	**30.48**	**0.41**	**30.48**	**0.27**
128	Cameraman	29.66	0.69	29.29	0.53	28.31	0.50	28.31	0.30
	Baboon	35.87	0.69	35.56	0.53	35.35	0.50	35.35	0.41
	Boats	30.27	0.69	29.98	0.53	29.09	0.50	29.09	0.35
	Bridge	27.75	0.69	27.54	0.53	27.24	0.50	27.24	0.41
	Lena	32.71	0.69	32.37	0.53	31.80	0.50	31.80	0.33
	Kush	32.99	0.69	32.73	0.53	32.31	0.50	32.31	0.36
	Goldhill	35.81	0.69	35.57	0.53	35.21	0.50	35.21	0.36
	Average	**32.15**	**0.69**	**31.86**	**0.53**	**31.33**	**0.50**	**31.33**	**0.36**
256	Cameraman	30.92	1.00	30.41	0.69	28.21	0.63	28.21	0.44
	Baboon	37.16	1.00	36.69	0.69	36.34	0.63	36.34	0.56
	Boats	31.36	1.00	30.94	0.69	29.64	0.63	29.64	0.47
	Bridge	28.60	1.00	28.32	0.69	27.66	0.63	27.66	0.54
	Lena	33.97	1.00	33.42	0.69	32.62	0.63	32.62	0.46
	Kush	34.29	1.00	33.81	0.69	33.15	0.63	33.15	0.49
	Goldhill	37.60	1.00	37.06	0.69	36.55	0.63	36.55	0.49
	Average	**33.41**	**1.00**	**32.95**	**0.69**	**32.02**	**0.63**	**32.02**	**0.49**
512	Cameraman	32.13	1.56	31.44	0.94	29.26	0.81	29.26	0.66
	Baboon	38.65	1.56	37.87	0.94	37.34	0.81	37.34	0.77
	Boats	32.39	1.56	31.82	0.94	30.12	0.81	30.12	0.68
	Bridge	29.60	1.56	29.19	0.94	28.20	0.81	28.20	0.75
	Lena	35.37	1.56	34.57	0.94	33.16	0.81	33.16	0.68
	Kush	35.53	1.56	34.92	0.94	33.97	0.81	33.97	0.71
	Goldhill	39.32	1.56	38.39	0.94	37.55	0.81	37.55	0.70
	Average	**34.71**	**1.56**	**34.03**	**0.94**	**32.80**	**0.81**	**32.80**	**0.71**
1024	Cameraman	34.63	2.63	33.29	1.38	29.84	1.13	29.84	1.01
	Baboon	40.60	2.63	39.31	1.38	38.42	1.13	38.42	1.09
	Boats	34.42	2.63	33.38	1.38	30.79	1.13	30.79	1.00
	Bridge	31.09	2.63	30.43	1.38	28.74	1.13	28.74	1.08
	Lena	37.66	2.63	36.32	1.38	33.94	1.13	33.94	1.02
	Kush	37.30	2.63	36.33	1.38	34.80	1.13	34.80	1.05
	Goldhill	41.61	2.63	39.97	1.38	38.69	1.13	38.69	1.04
	Average	**36.76**	**2.63**	**35.56**	**1.38**	**33.60**	**1.13**	**33.60**	**1.04**

TABLE III: AVERAGE PSNR, BPP AND COMPRESSION RATE (%) VALUES FOR DIFFERENT CODEBOOK SIZES

CB Size	CBEC			MMSE			Interpolative			SOC		
	PSNR	bpp	CR%	PSNR	bpp	CR%	PSNR	bpp	CR%	PSNR	Bpp	CR%
64	30.98	0.50	93.75	30.78	0.42	94.73	30.48	0.41	94.92	30.48	0.27	96.21
128	32.15	0.69	91.41	31.86	0.53	93.36	31.33	0.50	93.75	31.33	0.36	95.47
256	33.41	1.00	87.50	32.95	0.69	91.41	32.02	0.63	92.19	32.02	0.49	93.78
512	34.71	1.56	80.47	34.03	0.94	88.28	32.80	0.81	89.84	32.80	0.71	91.06
1024	36.76	2.63	67.19	35.56	1.38	82.81	33.60	1.13	85.94	33.60	1.04	86.88
Average			84.06			90.12			91.33			92.68

* CR% - Compression Rate in percentage

TABLE IV: PSNR AND BPP OBTAINED WITH THE EXISTING TECHNIQUES AND THE PROPOSED SCHEME.

CB Size	LBG		KPE		SCG		OCG		CBC		TSP		PM	
	PSNR	bpp	PSNR	bpp	PSNR	bpp	PSNR	bpp	PSNR	bpp	PSNR	bpp	PSNR	Bpp
64	31.01	0.50	30.99	0.50	31.05	0.50	30.96	0.50	31.02	0.44	24.31	0.50	30.48	0.27
128	32.15	0.69	32.14	0.69	31.87	0.69	32.11	0.69	31.98	0.56	25.82	0.69	31.33	0.36
256	33.42	1.00	33.36	1.00	33.13	1.00	33.13	1.00	33.11	0.75	27.66	1.00	32.02	0.49
512	34.90	1.56	34.87	1.56	34.44	1.56	34.44	1.56	34.40	1.06	28.84	1.56	32.80	0.71
1024	36.92	2.63	36.87	2.63	36.17	2.63	36.26	2.63	35.98	1.63	29.97	2.63	33.60	1.04

REFERENCES

[1] E. J. Delp and O. Robert, "*Image Compression using Block Truncation Coding*", IEEE Transactions on Communication, Col. COM-27, No. 9, 1335-1342, 1979.

[2] N. M. Nasrabadi and R. A. King, "*A New Image Coding Technique using Transform Vector Quantization*", Proceedings of International Conference on Acoustics, Speech and Signal Processing, San Diego, CA, pp. 29.9.1 – 29.9.4., 1984.

[3] J. S. Pan, Z. M. Lu, and S. H. Sun, "*An Efficient Encoding Algorithm for Vector Quantization Based on Subvector Technique*", IEEE Transactions on Image Processing, Vol 12, No. 3, March 2003.

[4] R. M.Gray, "*Vector Quantization*", IEEE Acoustics, Speech and Signal Processing Magazine, pp. 4-29, April 1984.

[5] Y. Linde, A. Buzo, and R. M. Gray, "*An Algorithm for Vector Quantizer Design*", IEEE Transactions on Communication", Vol. COM-28, No. 1, pp. 84-95, 1980.

[6] J. Z. C. Lai, Y. C. Liaw, and J. Liu, " *A Fast VQ Codebook Generation Algorithm using Codeword Displacement*", Pattern Recognition, Vol. 41, No. 1, pp. 315-319, 2008.

[7] Y. C. Liaw, J. Z. C. Lai, and W. Lo, "*Image Restoration of Compressed Image using Classified Vector Quantization*", Pattern Recognition, Vol. 35, No. 2, pp. 181-192, 2002.

[8] N. M. Nasrabadi and Y. Feng, "*Image Compression using Address Vector Quantization", IEEE Transactions on Communication, Vol.* 38, No. 12, pp. 2166-2173, 1990.

[9] J. Foster, R. M. Gray, and M. O. Dunham, "*Finite State Vector Quantization for Waveform Coding*", IEEE Transactions on Information Theory, Vol. 31, No. 3, pp. 348-359, 1985.

[10] T. Kim, "*Side Match and Overlap Match Vector Quantizers for Images*", IEEE Transactions on Image Processing, Vol. 1, No. 2, pp. 170-185, 1992.

[11] J. Z. C. Lai, Y. C. Liaw, W. Lo, "*Artifact Reduction of JPEG Coded Images using Mean-Removed Classified Vector Quantization*", Signal Processing, Vol. 82, No. 10, pp. 1375-1388, 2002.

[12] C. C. I. Katsavounidis, J. Kuo, and Z. Zhang, "*A New Initialization Technique for Generalized Lloyd Iteration*", IEEE *Signal Processing Letters, Vol. 1, No. 10, October 1994.*

[13] Gersho and R. M. Gray, "*Vector Quantization and Signal Compression" Dordrecht, The Netherlands: Kluwer, 1992.*

[14] K. Somasundaram and S. Vimala, "*Fast Codebook Generation for Quantization using Ordered Pairwise Nearest Neighbor with Multiple Merging*", In Proceedings of the IEEE International Conference on Emerging Trends in Electrical and Computer Technology (ICETECT), pp. 581-588, 2011.

[15] X. Wu and K. Zhang, "*A Better Tree Structured Vector Quantizer*", IEEE Proceedings of the Data Compression Conference, Snowbrid, UT, pp. 392-401, 1991.

[16] P. Franti, T. Kaukoranta and O. Nevalainen, "*On the Splitting Method for VQ Codebook Generation*", optical Engineering, Vol. 36, pp. 3043-3051, Nov' 1997.

[17] W. Equitz, "*A New Vector Quantization Clustering Algorithm*", IEEE Transactions Acoustics, Speech and Signal Processing, Vol. 37, No. 10, October 1989.

[18] A. Gersho and V. Cuperman, "*Vector Quantization: A pattern-matching technique for speech coding*," IEEE Communications Magazine, pp.15-21, 1983.

[19] K. Somasundram, I. Kaspar Raj, "*An Image compression Scheme based on Predictive and Interpolative Absolute Moment Block Truncation Coding*", GVIP Journal, Volume 6, Issue 4, December 2006

[20] P. Franti, O. Nevalainen, and T. Kaukoranta,"Compression of Digital Images by Block Truncation Coding:A Survey", The Computer Journal, Vol.37, No.4, 1994.

[21] S. P. Lloyd, 1982: "*Least Square Quantization in PCM*", IEEE Transactions on Information and Theory, Vol. 28, No. 2, pp. 129-137.

[22] H. B. Kekre and T. K. Sarode, "*Fast Codebook Search Algorithm for Vector Quantization using Sorting Technique*", Proceedings of International Conference on Advances in Computing, Communication and Control (ICAC3), 2009.

[23] K. Somasundaram and S. Vimala, "*Simple and Fast Ordered Codebook for Vector Quantization*", Proceedings of the National Conference on Image Processing (NCIMP 2010) held on Gandhigram Rural University, India, Allied Publishers Pvt. Ltd., pp. 3-7, 2010.

[24] S. Vimala, "*Low Bit-Rate Vector Quantization with Codebook Compression and Index Compression",* Proceedings of the International Conference on Computing: Upgrades and Trends (CUT 2010), APH Publishing Corporation, New Delhi, Chapter-22, pp. 285-298, 2010.

[25] S. Vimala, "*Techniques for Generating Initial Codebook for Vector quantization*", Proceedings of the International Conference on Electronics Computer Technology (ICECT 2011), IEEE Catalog No.: CFP1195F-ART, No. 4, pp. 201-208, 2011.