IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received June 17, 2021, accepted June 27, 2021, date of publication July 7, 2021, date of current version July 19, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3095287

Fast Codebook Generation Using Pattern Based Masking Algorithm for Image Compression

MUHAMMAD BILAL[1], ZAHID ULLAH[2], (Member, IEEE), AND IHTESHAM UL ISLAM[3]
[1]Department of Electrical Engineering, CECOS University of IT and Emerging Sciences, Peshawar 25000, Pakistan
[2]Pak-Austria Fachhochschule: Institute of Applied Sciences and Technology, Haripur 22650, Pakistan
[3]Military College of Signals, National University of Sciences and Technology, Islamabad 44000, Pakistan

Corresponding author: Zahid Ullah (zahid.ullah@fecid.paf-iast.edu.pk)

ABSTRACT Vector Quantization (VQ) is a classical block coding technique used for image compression which achieves high compression using simple encoding and decoding process. Codebook generation is an important factor in VQ design, which directly influences computational cost and the quality of the reconstructed image. Linde-Buzo-Gray (LBG) is considered as a state of art technique, which uses k-mean clustering algorithm for codebook design. Various optimization techniques are applied for searching the optimal codebook, such as Bat Algorithm (BA), Particle swarm optimization (PSO), and Firefly Algorithm (FA). These algorithm suffers mainly with high time consumption due to unavailability of the optimal solution in search space. This research proposes a novel approach, where peak values of the histogram are applied to predefined pattern masks to predict the image patterns for codebook design. From the experimental results, it is indicated that when compared with other algorithms, the proposed pattern based masking (PBM) algorithm requires fewer iterations and converges at a faster speed, particularly at the bitrates ≥ 0.375 without compromising on peak signal to noise ratio (PSNR) and structural similarity index measure (SSIM).

INDEX TERMS Computational time, codebook, image compression, LBG, peak signal to noise ratio, structure similarity index measure, vector quantization.

I. INTRODUCTION

Image compression has a key part in digital image processing by reducing the storage requirements and efficiently transmitting the compressed images. It is widely used in medical sciences, television transmissions, military applications, satellite communications, mobile applications and several other applications [1]. Mathematical optimization and efficient coding algorithms are generally used for various useful applications in computer science, image processing and robotics [2], [3]. These techniques are also employed for image and video compression.

Numerous techniques were proposed in the past such as predictive coding, transform-based coding and VQ for efficient image compression. VQ based image compression methods are considered to be the popular techniques for providing high compression ratio with less distortion compared to others [4]. The distortion can be adjusted according to application requirements by altering the block size [5]. VQ has a quick decompression mechanism which is suitable in such applications where images are decompressed more than one time such as multimedia and websites. Optimal codebook generation is one of the important step of the VQ which can be optimized using different optimization techniques including genetic algorithm [6], ant colony optimization [7], adaptive vector quantization [8] and other various techniques. VQ works by dividing the image into non-overlapping blocks named as input vectors. These input vectors are dependent upon the input image size and block size. The number of input vectors can be calculated by dividing the input image size by block size. For compression, quantization of the input vectors is performed to reduce the number of vectors required to represent the image. Selective input vectors (codewords) along with its indexes are transmitted as a compressed image. Several algorithms were presented in the literature to find the optimal codewords for codebook generation, such as Linde-Buzo-Gray (LBG) Algorithm which generates iteratively a codebook of size N from an initial randomly selected codebook [9]. It divides the initial vectors into groups of N clusters and refine the codebook in each iteration until the distortion is within acceptable limits. However, the algorithm gets trapped in local optima while

The associate editor coordinating the review of this manuscript and approving it for publication was Miaohui Wang.

estimating the distortion between the codebook and training vectors. Several optimization techniques have been proposed to improve the LBG algorithm. Huang and Xie [10] enhanced the algorithm of LBG by resolving the local optimal problem by adjusting the low utility and high utility codewords.

Rajpoot *et al.* [11] applied ant colony algorithm for the optimization of codebook to enhance the quality of reconstruction by using vector coefficients in 2-dimensional graph. It uses a set of ants as an agent to position the starting point in a search space. The ants cooperate with another for searching for better solutions using pheromones which is deposited on the edge of the graphs.

M. Kumar applied particle swarm optimization (PSO) to optimize the codebook generation by using global best and local best solutions [12]. Feng *et al.* [13] reported better performance in codebook design using fuzzy particle swarm optimization for searching global best solutions. Wang *et al.* [14] used quantum particle swarm optimization (QPSO) which outperformed PSO by calculation of local points and updating the position of particles for searching local and global best solutions. Horng and Jiang [15] proposed honey bee optimization mating optimization algorithm (HBMO) to achieve improvement in results compared with the previous techniques. Moreover, the application of swarm intelligence based firefly algorithm (FA) for optimizing codebook demonstrated the better quality of reconstructed image [16].

Multi-vector quantization (MVQ) uses two combinations of codewords for codebook design. The codewords and the index mappings are separately coded which provides an efficient compression of hyperspectral imagery (HSI) [17]. Hosseini and Nighsh-Nilchi [18] proposed contextual vector quantization (CVQ) focused on contextual information of an image. Such information contains essential part of an image, which must be encoded with a high priority compared to non-contextual information. Tsolakis *et al.* [19] applied fast fuzzy vector quantization by removing the codewords which are affected by particular training vectors. The number of those training vectors are reduced which are moving from smaller to larger clusters using code vectors migration strategy. Karri and Jena [20] applied bat algorithm to improve the quality of the reconstructed image by adjusting loudness and pulse emission rate of the bats. Feng *et al.* [21] proposed edge oriented patterns (EOPs) by defining nine different classes of codebooks. The algorithm uses LBG for codebook generation, however, the edges are treated separately. The edge orientation is determined using edge templates and they are compressed using a specific class of codebook. The reconstructed image shows improvement in edge degradation. Chiranjeevi and Jena [22] applied a Cuckoo search (CS) optimization for codebook design using the levy flight distribution function to enhance the quality of the reconstructed image.

Various optimization techniques have been employed to optimize the codebook generation process; however, altogether LBG based VQ schemes provide improvement in quality but the algorithm converges with a slower speed. Therefore, in this work, a fast LBG based algorithm is proposed which uses heuristic approach for codebook generation.

A. KEY CONTRIBUTIONS

The proposed research work has the following key contributions.

- The proposed PBM algorithm outperforms the conventional LBG and LBG based algorithms in terms of computational time at the bit rates ≥ 0.375.
- The variation of average peak signal to noise ratio with respect to bit rate shows the proposed PBM algorithm attains a better PSNR with reasonable time.
- The algorithm achieved improved PSNR and SSIM compared to LBG.

This paper contains five sections which includes an introduction. Section 2 contains the contemporary algorithm for the codebook design along with their algorithm. In section 3 the proposed methodology is elaborated. Results and discussion is given in section 4. The conclusion and future work is presented in section 5.

II. CONTEMPORARY ALGORITHMS FOR CODEBOOK DESIGN

VQ is a block coding technique in which codebook generation is an essential part. The encoding and decoding of VQ is depicted in Fig 1 where a test image of size (N *x* N) is divided in to smaller block of size *Nb* (n *x* n). These blocks are training vectors and they are represented as *Xi* (i $= 1, 2, 3, 4 \ldots\ldots Nb$). The codebook is reconstructed from the particular vectors from the Nb blocks. The selected vectors are the codewords of codebook and they are represented as *Ci* (i $= 1, 2, 3 \ldots\ldots Nc$) where *Nc* is the total codewords in the codebook. Image vectors are estimated through the index of the codewords, which is approximated by finding the minimum value of Euclidean distance between the image vectors and the codewords. The final index and codebook are transmitted to the receiver, where it is used to reconstruct the image through codewords and corresponding indexes. The distortion between the codebook and training vector is calculated as

$$D = 1/N_c \sum_{J=1}^{Nc} \sum_{i=1}^{Nb} u_{ij} * ||X_i - C_j||^2 \tag{1}$$

With subject to constraint

$$D = \sum_{J=1}^{Nc} u_{ij} = 1, \quad \forall i \epsilon \{1, 2, \ldots, N_b\} \tag{2}$$

$$u_{ij} = \begin{cases} 1, & X_i \epsilon J^{th}\ Cluster \\ 0, & Otherwise \end{cases} \tag{3}$$

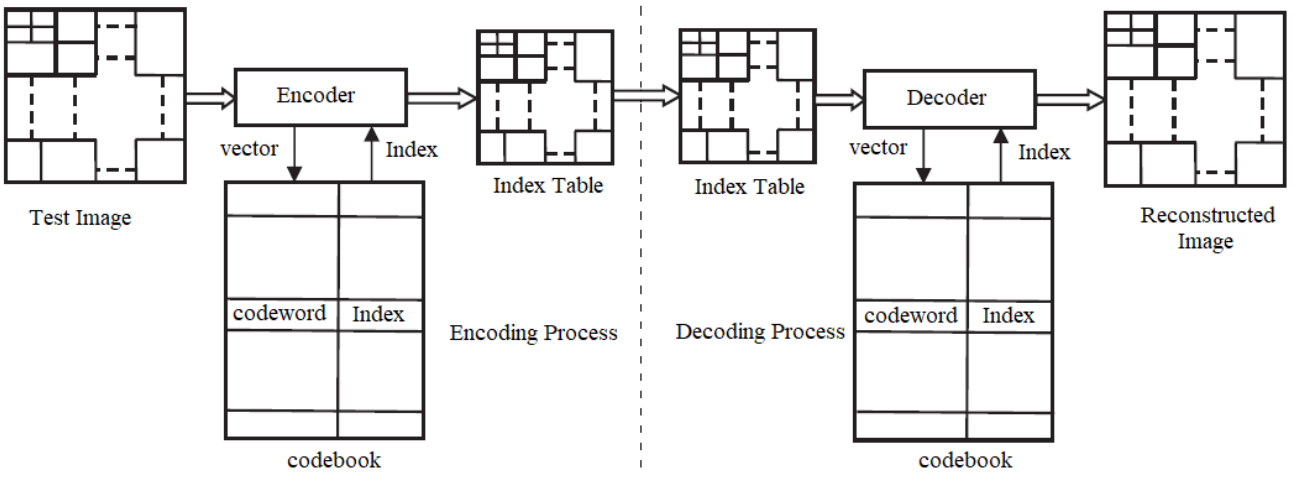

FIGURE 1. Encoding and decoding using vector quantization.

For vector quantization two conditions are necessary

(1) The partition $R_j \forall j = 1, 2, 3 \ldots, N_c$ should meet the criterion

$$R_j \supset \{x \epsilon X : d(c, C_j) < d(x, C_k), \quad \forall k \neq j \tag{4}$$

(2) The codeword C_j should be defined as the centroid of R_j such that

$$C_j = 1/N_j \sum_{i=1}^{Nj} x_i, \quad \forall x_i \epsilon R_j \tag{5}$$

where, N_j is the total number of vectors that belongs to R_j

A. LBG VECTOR QUANTIZATION ALGORITHM

Generalized loyd algorithm (GLA) is also recognized as LBG algorithm is pioneer algorithm for the implementation of VQ [9]. Algorithm 1 shows the LBG VQ. It is a k-mean based clustering algorithm that tries to find the local optimum solution using the closest match function. This function tries to ensures that distortion is not increased from one iteration to the next. However, this approach has a limitation of getting trapped in local optima due to its inefficient random initialization of the initial codebook.

B. PSO VECTOR QUANTIZATION ALGORITHM

The Particle Swarm Optimization (PSO) was proposed by Hsuan and Ching in 2007. The algorithm is based on the principles of natural behavior of school fishing and bird flocking in which the codebook adjusts the codewords (particles) from previous values using swarm intelligence, which is depicted in Algorithm 2. The algorithm provides a global codebook but consumes high iteration if the particle update velocities is high.

C. QPSO VECTOR QUANTIZATION ALGORITHM

Wang *et al.* [14] proposed another novel Quantum Swarm Evolutionary Algorithm (QPSO) as elaborated in Algorithm 3

Algorithm 1: LBG Algorithm

Initialize C_1 as initial codebook and assign training set of cluster using R_i

Initialize a counter m=1 and Set distortion D1 = α;

while $D_m - D_{m+1} < T$ **do**

- Assign codebook $C_m = Y_i$, where Y_i are selected randomly from training set.
- Calculate the centroids using eq (5) through R_i.
- Assign it as a refined codebook C_{m+1}.
- Measure the distortion using eq (4).

If ($D_m - D_{m+1} >$ T), stop execution.
else m=m+1;

end

which estimates the local points from local (pbest) and global best (gbest) codebook as P_i using equation 10. The u and z parameters are used for updating the position of the particles. These parameters require fine tuning to gain improvement in terms of PSNR compared to LBG and PSO-LBG but consume high computation.

$$P_i = r_1 pbest_i + r_2 gbest_i / r_1 + r_2 \tag{10}$$

D. FA VECTOR QUANTIZATION ALGORITHM

Horng [16] proposed firefly's algorithm for codebook design. It is based upon the flashing phenomena, in which the objective function is dependent on the brightness of fireflies. Multiple codebooks (fireflies) are initialized in which each codebook tries to move from the lower intensity to higher intensity. However, the algorithm suffers in PSNR if there are no brighter flies in the search space. The details of FA-VQ is depicted in Algorithm 4.

E. BA VECTOR QUANTIZATION ALGORITHM

Bat algorithm for vector quantization was proposed which works on the mating principle of bats [20]. The codebook

Algorithm 2: PSO-LBG Algorithm

Execute LBG algorithm; allocate the codebook as global best codebook (gbest)
Initialize codebooks using random velocities, assign max iteration count = j
Set count m=1
while *(m < j)* **do**

- Calculate the fitness function of each code book using.

$$Fitness(C) = \frac{1}{D(C)} \tag{6}$$

$$\frac{1}{D(C)} = \frac{N_b}{\sum_{J=1}^{Nc}\sum_{i=1}^{Nb} u_{ij} * ||X_i - C_j||^2} \tag{7}$$

where $D(C)$ is distortion,X_i is the vector of training set and C_j represents the codeword of the codebook.

- If fitness value is lower than previous fitness, allocate new fitness as p_{best}.
- Select highest fitness and replace it with g_{best}.
- Velocities and positions of each particle is updated using following equations.

$$V_{ik}^{n+1} = V_{ik}^{n} + C_1 r_1^n (pbest_{ik}^n - X_{ik}^n) + C_2 r_2^n (gbest_k^n - X_{ik}^n) \tag{8}$$

$$X_{ik}^{n+1} = X_{ik}^{n} + V_{ik}^{n+1} \tag{9}$$

Here total number of solutions are k, where i is particle position,r_1 and r_2 are random number where as C_1 and C_2 are social and cognitives rates.

- If (m=j) execution stops
- Else m=m+1;

end

Algorithm 3: QPSO-LBG Algorithm

Initialize the Loy'ds (LBG) algorithm;
Allocate it as the global best codebook (g_{best}).

- Initiate other codebooks using random velocities.
- Assign maximum iteration count = j
- Set count m=1

while *(m < j)* **do**

- Calculate the fitness of all codebooks using Eq(6).
- Incase the fitness is improved compare to previous value (P_{best}), assign it as (P_{best}).
- Find the maximum value of fitness in all particles
- In case if found improvement compared to g_{best}, then g_{best} is replaced with new value.
- Chose random r_1, r_2 and u between the ranges of 0 to 1
- Calculate the local points P_i using Eq(9)
- Elements of the codebook X_i are updated as under

$$L_i = z|X_i - p_i| \tag{11}$$

$$if\ u > 0.5 X_i(t+1) = p_i - L_i * ln(1/u) \tag{12}$$

$$else\ X_i(t+1) = p_i + L_i * ln(1/u) \tag{13}$$

Here z is a constant and it is kept as $z < 1/ln\sqrt{2}$ where t shows the iterations

- If (m=j) execution stops
- Else m=m+1;

end

is considered as bat, which uses three tuning parameters loudness, frequency and pulse rate for the estimation of global codebook. The BA-LBG algorithm attains high PSNR compare to other techniques such as LBG, PSO-LBG, HBMO-LBG and FA-LBG. The algorithm requires the calculation of the additional parameter, which consumes high computation time compared to QPSO, PSO and LBG algorithms. The details of BA-VQ is described in Algorithm 5.

F. CS VECTOR QUANTIZATION ALGORITHM

Chiranjeevi and Jena [22] proposed cuckoo search LBG vector quantization. The cuckoo search works on the principle of nest/egg which is treated as a codebook. It is a metaheuristic algorithm, which uses mutational probability to find the global codebook. Improvement is observed in CS algorithm in terms of image quality. The algorithm requires the recursive computation of new nest from the old one causing high computation. The working principle is discussed in the Algorithm 6.

III. PROPOSED PBM-LBG ALGORITHM

The conventional LBG has a limitation as it gets stuck in local optima due to poor random initialization of the initial codebook. PSO and QPSO achieve efficient codebook but suffers instability for particles with higher velocity. HBMO requires several tuning parameters in codebook design [23]. The FA suffers in convergence if search space contains no brighter fireflies [24]. CS algorithm undergoes high iteration if it does not meet convergence criteria [25]. To reduce computational time and maintain high PSNR, a new approach is adopted to modify the LBG using histogram oriented pattern-based masking (PBM). The block diagram of the proposed method is depicted in Fig 2.

The histogram of the test images was plotted to find the curve, which shows the tonal density over the entire spectrum [26]. The histogram of an 8-bit grayscale image can take 256 distinct pixel values (grey levels) along with frequency distribution (pixel count). The distribution from 0-255 pixel values are numbered from dark to bright pixels. The histogram of test image 'Lena' is shown in the Fig 3, which contains 185 grey levels along with the frequency distribution of each gray level. The peaks tonal values within a certain threshold are extracted from the histogram as they contains

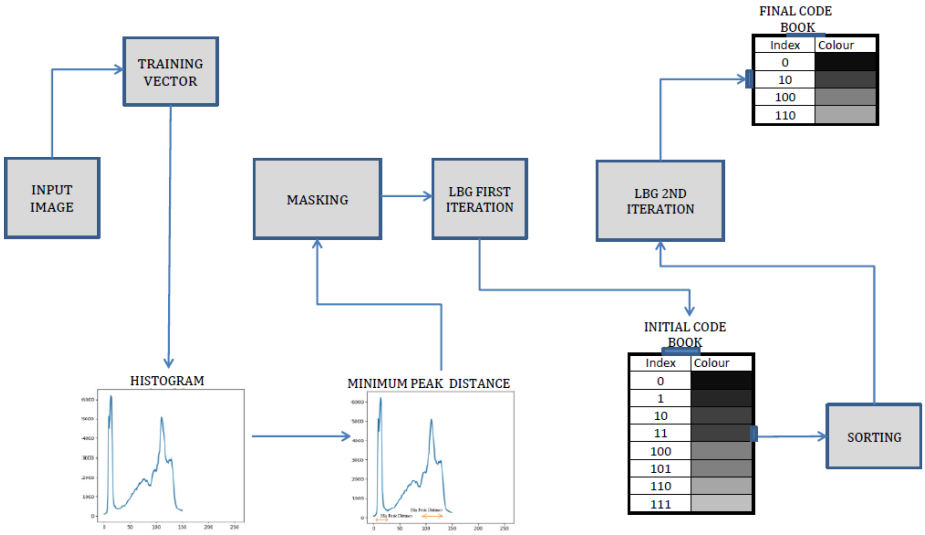

FIGURE 2. Block diagram of proposed pattern based masking algorithm.

Algorithm 4: FA-LBG Algorithm

Initialize loy'ds (LBG) algorithm and assign it as the bright firefly (codebook).
Initiate the parameters alpha (α) Beta (β) and gamma coefficients (λ).
Initialize codebook randomly, assign maximum iteration count = j and set count m=1
while $(m < j)$ **do**

- Calculate the fitness of all the codebook using Eq(6).
- Choose a random codebook and measure its fitness.
- Move codebook towards the bright fireflies using equations (17)-(19)

$$\textit{Euclidean distance } r_{ij} = ||X_i - X_j|| \tag{14}$$

$$||X_i - X_j|| = \sqrt{\sum_{k=1}^{N_c}\sum_{h=1}^{L}(X_{ik}^h - X_{jk}^h)^2} \tag{15}$$

$$\beta = \beta_o e^{-\gamma ij} \tag{16}$$

here 0<u<1 where k = (1, 2, 3, ...N_c)

- If no brighter flies are found, then randomly move in search space using

$$X_{jk}^h = (1-\beta)X_{ik}^h + \beta X_{jk}^h + u_{jk}^h \tag{17}$$

- If (m=j) execution stops
- Else m=m+1;

end

a high contribution of pixel values in the test images. The histogram is analyzed as a continuous signal in which peaks

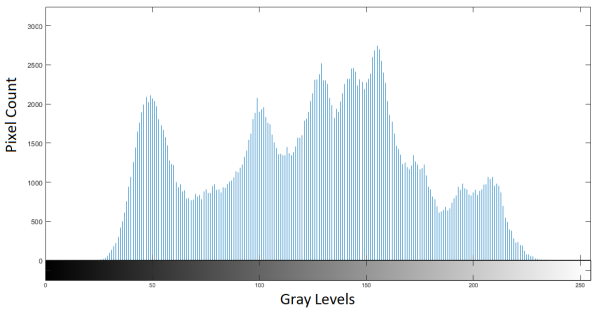

FIGURE 3. Lena histogram.

(local maxima) from the graph [27] are used as suitable values for pattern estimation. The peaks values from histogram are measured using a derivative function where the intensity of the signal is zero with respect to pixel value by using equation.

$$P(i) = h(i) \quad \forall \frac{d}{dx} i = 0 \tag{25}$$

Here, P_i is the peak value and h_i is the histogram of the signal. If a selected range of peaks separation is small, it can cause codeword replication problem. Moreover, it creates large codewords entries in the initial codebook. If peaks separation is kept high, it will produce very few peaks and might create an inefficient codebook causing poor reconstruction in decoding process, hence for pattern estimation only one peak (local maxima) is selected within a nominal range.

A pattern shows regularity in an image which can be identified as a combination of progression in direction of tonal densities. A block of an image can take different forms of patterns which may contains a flat, a slope or an edge of an image [28]. Histogram values are used to predict the correlation between the neighboring pixels by finding the most frequently occurred blocks. The variation of tonal distribution is predicted to a certain degree when analyzing a

Algorithm 5: BA-LBG Algorithm

Initialize the following, N code books (bats), A (Loudness), V (Velocity), R (pulserate), Qmin (Minimum frequency) and Qmax (Maximum frequency).
Run the Loy'ds (LBG) and allocate it as an initial codebook
The other codebooks X_i (i = 1, 2, 3, ...N − 1) are randomly chosen.
Assign maximum iteration count = j and set count m=1
while *(m < j)* **do**

- Calculate the fitness of all code books using Eq(6). Assign *Xbst* to the best codebook
- The codebook is updated through assigning new position, frequency and velocity using Eq(20)-(22)

$$Q_1(t+1) = Q_{max}(t) + (Q_{min}(t) - Q_{max}(t)) * (R) \tag{18}$$

$$V_1(t+1) = V_i(t) + (X_i(t) - X_{best}(t)) * Q_i(t+1) \tag{19}$$

$$X_1(t+1) = X_i(t) + V_i(t) \tag{20}$$

- The step size is generated between 0-1 for the random walk(W).
- If step size is higher than the R (pulse rate) then move the codebook using Eq(23)

$$X_1(t+1) = X_{best}(t) + W * R \tag{21}$$

- Randomly generate a value if it lower than loudness, include the codebook.
- Sort the codebooks according to their fitness value X_{best}.
- If (m=j) execution stops
- Else m=m+1;

end

Algorithm 6: CS-LBG Algorithm

Initiate the Loy'ds(LBG) algorithm and allocate it as codebook(nest)
Randomly initialize remaining codebooks (nest).
Assign maximum iteration count = j
Set count m=1
while *(m < j)* **do**

- Measure fitness value of the nests and assign maximum as $nest_{best}$
- Produce new nest using Levy random walk using equation.

$$nest_{new} = nest_{old} + \alpha \bigotimes Le'vy(\lambda) \tag{22}$$

- Generate a random number K.
- If mutation probability (Pa) is less than K, swap worst nest with new one.
- Keep new nest and generate new nest using random step size as

$$nest_{new} = nest_{old} + (K + stepsize) \tag{23}$$

where

$$stepsize = r * (nest_{rand} + nest_{rand}) \tag{24}$$

where r is a random number

- Perform ranking the nests on basis of fitness function using Eq(6)
- Select the best fitness as $nest_{best}$
- If (m=j) execution stops
- Else m=m+1;

end

small group of pixels, this variation is either a single tone distributed with a minor variation or a linear progression of tones from one pixel to another. This variation can also be non-linear which can be noticed over edges where drastic changes occur in tonal variation in the neighboring pixels. We have designed 64-patterns masks of 4 × 4 pixels with linear and non-linear tonal progression which provide several combinations of patterns covering the repetition or slight change of tonal values as well as the drastic change in tonal values over the edges. The number of masks incorporated can be of an indefinite number, however, increasing the number of masks would improve the quality of image at the expense of high processing [29]. A pattern can be applied to any peak value from the histogram, but practically not all the patterns are suitable for every selected peak due to the boundary constraints of 256 maximum value. Since the pixel values must remain between 0-255, hence a pattern with a large slope of progression cannot be applied to a very high peak value. All of 64 pattern masks are added with peak values selected from the histogram in order to adjust the mask for the prediction of best pattern match to the training vector. Eq(26) is used for the estimation of patterns.

$$C_i = P_i + \sum_{j=0}^{63} M_j \quad \forall P_i + M_i < 255 \tag{26}$$

where, *Ci* represents the i^{th} codeword of initial codebook, *Pi* is the i^{th} peak selected from the histogram and *Mi* shows the mask pattern.

The first iteration is performed by finding the closest match between the *Ci* patterns with the training vector. Once the iteration is performed all the codebook patterns are placed in a single temporary codebook. The initial codebook patterns are large in number which needs to be reduced to the desired size. A quick sort is performed to arrange the codebook on the basis of ascending order from most to least repeated patterns. The most repeated patterns are selected to arrange the codebook to the desired size. However, a 2nd iteration is required

Algorithm 7: PBM-LBG Algorithm

- Initialize (I=$I_1, I_2, I_3 \ldots\ldots\ldots I_k$) as initial training vectors.
- Calculate the histogram of training vector with intensity value in range such that

$$L(u, v) \epsilon [0, K-1] \tag{27}$$

Would contain exactly K entries such that k=2^8=256 and each histogram entry is defined as

$$h(i) = card\{(u, v) \mid I(u, v) = i\} \tag{28}$$

- Estimate the peak (P) values of histogram using Eq(23)
- Initialize Mask (M) of 4*4 metrics of predefined 64 blocks of distinct patterns (p). Where $M_i = (P_1, P_2, P_3, \ldots\ldots P_{63})$
- Add peak (Pi) to each predefined mask (M) using Eq(24) Where p_i is the peak values in histogram, m_i is the Minimum peak distance and S is the separation between the peak values.
- Place all the M_i pattern blocks as an initial codebook CBi.
- Initialize m = 0.

while *(m < 2)* **do**

- Measure distortion between codewords in CBi and training vectors using Eq(1).
- Sort the initial code book and assign it to the final codebook of size Nb, such that each entry of final codebook (CBi +1) contains code words having closest match among training vector and initial codebook.
- If (m=j) execution stops
- Else m=m+1;

end

to update the index table and obtain a final codebook using closest match function (least Distortion) between the training vector and codebook using Eq(1). The finalized codebook along with the index table is transmitted, which is used by the receiver for decoding the test image. The proposed algorithm for vector quantization is depicted in Algorithm 7.

IV. RESULTS AND DISCUSSION

The traditional methods to evaluate the codebook designs are performed on grey scale images [30]. On the contrary, five different test images including 'Lena.jpg' with 185 gray levels, 'Baboon.png' with 199 gray levels, 'peppers.png' with 220 gray levels 'Barb.png' with 214 gray levels and 'Goldhill.png' with 213 gray levels were used for the comparative analysis. The simulations are performed using windows 10 pro edition with Intel (R) Core (TM) i5-3210 and 2.5 GHz CPU with 4 GB DDR3 RAM. The codes are compiled using MATLAB version (R2016A). All the tests are performed on the resolution of 512*512 grey scale image which are

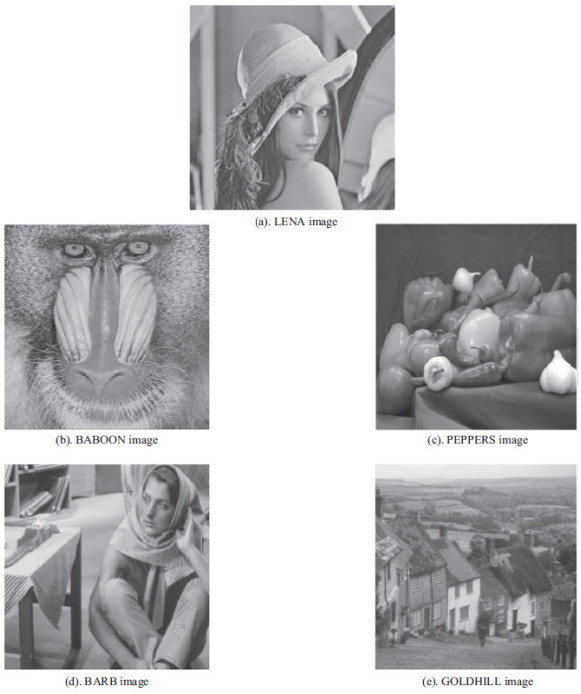

FIGURE 4. Test images: (a) Lena.jpg with 185 gray levels. (b) Baboon.png with 199 gray levels. (c) Peppers.png with 220 gray levels. (d) Barb.png with 214 gray levels. (e)Goldhill.png with 213 gray levels.

shown in the Fig 4.(a-e). All the test images are compressed using PBM, CS-LBG, FA-LBG, BA-LBG, HBMO-LBG, QPSO-LBG, PSO-LBG and LBG.

To perform compression, the test image is first subdivided into a non-overlapping block of size 4*4 pixels. These non-overlapping blocks are considered as a training vector of size 16384, where each input vector dimension is kept 16. The bit rate per pixel (bpp), Mean Square Error (MSE), and Peak Signal to Noise ratio (PSNR) are used as parameters for comparison, which are calculated using Eqs. 29, 30, and 31, respectively.

$$bpp = \frac{Log_2 N_c}{k} \tag{29}$$

where, k denotes the block size and N_c is codebook size. The bitrate per pixel is used for evaluating the compressed image size.

$$MSE = \frac{1}{M * N} \sum_{I}^{M} \sum_{J}^{N} \{f(I, J) - f^{-}(I, J)\}^2 \tag{30}$$

where M*N are total number of pixels. I and J are the x and y coordinates of pixel's values. The function f (I, J) represents the test image whereas f^{-}(I, J) shows the compressed image.

$$PSNR = 10\ log_{10}(\frac{255}{MSE})^2\ (db) \tag{31}$$

The quality of the decompressed image is evaluated using PSNR values. The PSNR of six test images were used having standard image size of M*N (512*512) pixels, and the codebook size (8, 16, 32, 64, 128, 256, 512 and 1024). The values

TABLE 1. Bitrate vs PSNR of test image 'Lena'.

Bitrate	Peak signal to Noise ratio (PSNR) in db						
	LBG	PSO -LBG	QPSO -LBG	HBMO -LBG	FA -LBG	BA -LBG	PBM
0.15	24.1	24.2	24.2	24.1	24.1	24.2	23.5
0.25	25.4	25.4	25.4	25.3	25.3	25.5	26.3
0.325	25.2	25.5	25.6	25.4	25.5	25.7	27.2
0.375	25.2	26.0	25.9	25.8	25.7	26.3	28.2
0.435	25.3	27.6	27.5	27.5	27.6	28.0	29.2
0.485	25.4	28.5	28.6	28.6	28.7	28.8	29.6
0.55	25.7	29.0	29.1	29.2	29.2	29.5	29.7
0.625	25.7	30.5	30.6	30.7	30.7	30.8	29.8

TABLE 2. Bitrate vs PSNR of test image 'Baboon'.

Bitrate	Peak signal to Noise ratio (PSNR) in db						
	LBG	PSO -LBG	QPSO -LBG	HBMO -LBG	FA -LBG	BA -LBG	PBM
0.15	18.3	18.4	18.2	18.8	19	19.2	19.2
0.25	19.7	19.7	19.8	19.9	19.7	20.2	19.8
0.325	19.6	20.1	20	20.3	20.1	20.3	20.5
0.375	19.7	20.6	20.8	21.1	20.9	21.7	21.1
0.435	19.8	21.6	21.5	21.9	21.7	22.3	21.8
0.485	19.7	22.1	22.3	22.7	22.5	23	22.3
0.55	19.7	23.2	23.1	23.5	23.2	23.7	22.7
0.625	19.8	23.5	23.5	23.7	23.6	24.5	23.1

TABLE 3. Bitrate vs PSNR of test image 'Peppers'.

Bitrate	Peak signal to Noise ratio (PSNR) in db						
	LBG	PSO -LBG	QPSO -LBG	HBMO -LBG	FA -LBG	BA -LBG	PBM
0.15	24.3	24.4	24.5	24.5	24.5	24.7	24.1
0.25	25.2	25.4	25.5	25.3	25.1	25.6	24.8
0.325	25.3	26.3	26.5	26.4	26.2	26.4	26.2
0.375	25.3	27.2	27.3	27.5	27.7	28.5	26.8
0.435	25.3	29.3	29.5	29.6	29.8	30.1	27.8
0.485	25.4	30.2	30.4	30.5	30.6	30.8	29.1
0.55	25.4	31.2	31.4	31.5	31.6	31.8	29.7
0.625	25.4	32.5	32.6	32.7	32.8	32.6	30.1

TABLE 4. Bitrate vs PSNR of test image 'Barb'.

Bitrate	Peak signal to Noise ratio (PSNR) in db						
	LBG	PSO -LBG	QPSO -LBG	HBMO -LBG	FA -LBG	BA -LBG	PBM
0.15	23.8	24.1	23.7	23.5	23.5	24.3	23.2
0.25	23.8	24	24.2	24.2	24.2	24.6	24.2
0.325	24.0	25.8	25.8	25.8	26.3	26.3	26.3
0.375	24.2	27.2	27.3	27.2	27.3	27.6	27.5
0.435	24.0	28.2	28.4	28.3	28.5	28.4	28.1
0.485	24.8	28.8	28.8	28.9	29.0	29.0	28.6
0.55	24.6	30.0	30.1	29.8	29.9	30.1	28.9
0.625	25.0	30.1	30.2	29.8	29.9	30.9	29.3

TABLE 5. Bitrate vs PSNR of test image 'Gold hill'.

Bitrate	Peak signal to Noise ratio (PSNR) in db						
	LBG	PSO -LBG	QPSO -LBG	HBMO -LBG	FA -LBG	BA -LBG	PBM
0.15	24.5	24.6	24.3	24.3	24.3	24.9	23.8
0.25	25.2	25.3	25.3	25.6	25.8	25.9	24.8
0.325	25.6	25.3	25.9	25.9	26	26.2	26.2
0.375	25.7	26.2	26.2	26.9	26.8	27.0	27.2
0.435	25.7	27.3	27.4	27.7	27.8	28.1	28.1
0.485	25.7	28.1	28.2	28.8	28.7	28.9	28.9
0.55	25.7	29.8	29.9	30.2	30.2	30.1	29.3
0.625	25.7	30.2	30.2	30.5	30.4	30.9	29.8

of the parameters in PBM algorithm for simulation of the test image is measured on the basis of the maximum average values of the PSNR (five times). The parameter of the pattern mask is adjusted to 64 using hit and trial method which provides an optimized codebook. Through trails it was estimated that lowering the number of pattern masks improves the run time but avoids the useful patterns which drastically affect the PSNR value whereas increasing the number of mask pattern from 64 increases the processing time, as well as no significant improvement is detected in PSNR values. The PSNR comparison of test images using PBM is compared with the existing algorithms, are shown in the Table (1-5). The variation of average peak signal to noise ratio with respect to bit rate shows that the PBM algorithm attains a better PSNR with reasonable time. It can be noticed that in most of the test images, the proposed algorithm attains high PSNR compared to LBG at the lower bit rates particularly from the bit rate of $\geq$ 0.325 and provides approximately similar types of results for the bit rate 0.325 to 0.55 with respect LBG based algorithms.

The PSNR provides good estimation for quality comparison of images, however, it does not represents the best match for human visual perception, which is highly capable of identifying structural information [31]. To obtain structural comparison between the test image and compressed image, Structural Similarity Index measure (SSIM) metrics have been calculated for comparing luminance, contrast and structure between the two images. SSIM score is calculated using Eq(32).

$$SSIM(X, Y) = [L(X, Y)]^{\alpha}.[C(X, Y)]^{\beta}.[S(X, Y)]^{\gamma} \quad (32)$$

where, L, C, S represent luminance, contrast and structure respectively and α, β, γ denotes the relative importance of each of these parameters. For simplification, it was assumed that $\alpha = \beta = \gamma = 1$. The values for SSIM have been found to

TABLE 6. Average computational time using five test images.

Codebook size: 16								
Image	**Average computation time (sec) at bitrate = 0.25**							
	LBG	PSO-LBG	QPSO-LBG	HBMO-LBG	FA-LBG	BA-LBG	CS-LBG	PBM
LENA	7.74	572.83	607.39	1202.47	1152.89	579.72	2288.09	10.76
PEPPER	8.91	487.58	493.46	1105.29	1040.35	630.45	3326.91	11.33
BABOON	9.45	669.85	695.20	1983.13	1964.47	698.99	3031.07	10.73
GOLDHILL	9.65	625.38	740.90	1158.51	1130.76	513.29	2480.96	10.66
BARB	9.23	555.68	656.92	1567.50	1549.54	690.41	2811.52	9.88
Average	9.00	582.26	638.78	1403.38	1367.60	622.57	2787.71	10.67

Codebook size: 32								
Image	**Average computation time (sec) at bitrate = 0.3125**							
	LBG	PSO-LBG	QPSO-LBG	HBMO-LBG	FA-LBG	BA-LBG	CS-LBG	PBM
LENA	8.77	510.89	530.52	1268.91	1184.69	572.90	2194.91	11.20
PEPPER	9.81	532.18	428.91	898.77	934.77	546.92	1713.64	11.49
BABOON	8.89	468.13	497.97	1249.00	1243.72	549.35	2715.30	11.63
GOLDHILL	7.71	476.65	538.47	1340.22	1299.81	480.46	2625.01	10.77
BARB	10.04	423.94	474.93	1349	1320.16	422.79	2025.71	11.22
Average	9.04	482.36	494.16	1221.99	1196.63	514.48	2294.52	11.26

Codebook size: 64								
Image	**Average computation time (sec) at bitrate = 0.3750**							
	LBG	PSO-LBG	QPSO-LBG	HBMO-LBG	FA-LBG	BA-LBG	CS-LBG	PBM
LENA	10.28	645.61	663.61	1545.10	1474.84	661.69	2963.34	10.25
PEPPER	11.30	597.43	599.23	1247.55	1278.46	636.78	4468.24	11.64
BABOON	12.26	573.60	590.13	1412.33	1437.10	740.01	3984.19	12.59
GOLDHILL	14.33	622.23	637.76	1577.16	1181.35	498.58	4305.05	11.12
BARB	16.74	460.22	466.25	1306.20	854.18	398.75	2721.10	11.31
Average	12.98	579.82	591.40	1417.67	1244.82	587.16	3688.38	11.38

Codebook size: 128								
Image	**Average computation time (sec) at bitrate = 0.4375**							
	LBG	PSO-LBG	QPSO-LBG	HBMO-LBG	FA-LBG	BA-LBG	CS-LBG	PBM
LENA	14.16	622.05	636.07	1059.20	1031.52	613.59	1917.13	11.89
PEPPER	19.42	600.31	675.31	1132.69	1081.50	623.63	2220.83	11.71
BABOON	22.62	467.59	536.09	1112.15	1060.78	963.92	2785.67	12.12
GOLDHILL	18.22	835.28	860.39	1413.09	1343.74	502.22	1962.01	11.23
BARB	30.35	579.20	589.55	1291.80	1271.33	562.25	2438.00	12.11
Average	20.95	620.89	659.48	1201.79	1157.77	653.12	2264.73	11.81

Codebook size: 256								
Image	**Average computation time (sec) at bitrate = 0.50**							
	LBG	PSO-LBG	QPSO-LBG	HBMO-LBG	FA-LBG	BA-LBG	CS-LBG	PBM
LENA	21.14	885.45	904.33	808.03	797.70	672.26	1614.72	12.11
PEPPER	18.35	760.11	760.11	1010.66	984.55	574.42	1750.67	12.12
BABOON	28.26	599.50	568.25	1019.87	1040.92	572.31	2039.76	12.13
GOLDHILL	29.93	931.60	560.65	850.07	834.33	981.94	2978.07	12.33
BARB	27.85	689.70	698.67	847.24	837.71	596.07	2598.44	12.01
Average	25.11	773.27	698.40	907.17	899.04	679.40	2196.33	12.14

Codebook size: 512								
Image	**Average computation time (sec) at bitrate = 0.5625**							
	LBG	PSO-LBG	QPSO-LBG	HBMO-LBG	FA-LBG	BA-LBG	CS-LBG	PBM
LENA	34.65	711.81	736.05	1101.60	1056.29	826.00	1623.34	12.22
PEPPER	39.30	934.71	887.54	665.20	650.80	533.06	1371.77	12.12
BABOON	20.73	657.03	715.01	712.06	723.13	803.58	1158.43	12.33
GOLDHILL	35.15	582.98	601.75	955.59	960.06	885.01	1253.41	11.33
BARB	72.51	815.85	706.92	878.69	872.66	693.67	1805.60	11.22
Average	40.47	740.48	729.45	862.63	852.59	748.26	1442.51	11.84

TABLE 6. *(Continued.)* **Average computational time using five test images.**

Codebook size: 1024								
Image	**Average computation time (sec) at bitrate = 0.625**							
	LBG	PSO-LBG	QPSO-LBG	HBMO-LBG	FA-LBG	BA-LBG	CS-LBG	PBM
LENA	64.38	1512.15	1554.60	1895.57	1868.68	1551.37	3654.66	12.11
PEPPER	63.45	1022.79	1156.54	1254.01	1231.38	855.75	2059.36	12.12
BABOON	66.56	1435.46	1439.53	1664.24	1636.12	1665.14	2396.34	12.13
GOLDHILL	94.67	1353.84	1369.65	1489.71	1483.25	775.30	2340.92	12.33
BARB	112.31	1515.01	1503.80	1199.28	1181.89	1133.71	2400.29	12.22
Average	80.27	1367.85	1404.82	1500.56	1480.27	1196.25	2570.31	12.18

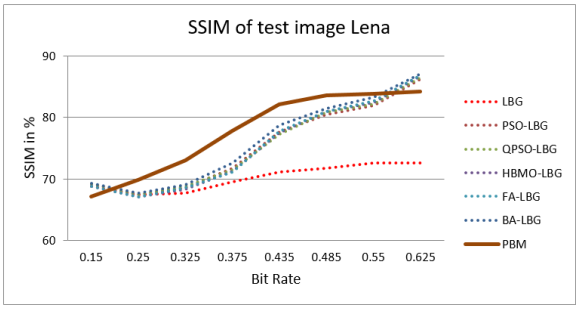

FIGURE 5. Lena SSIM.

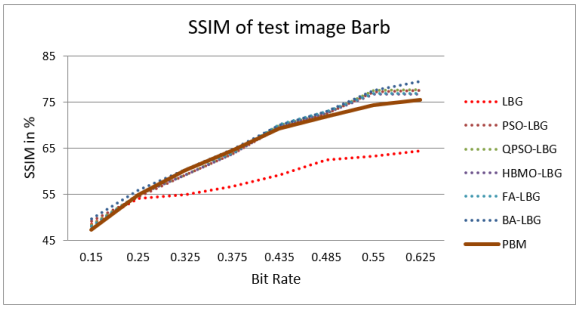

FIGURE 8. Barb SSIM.

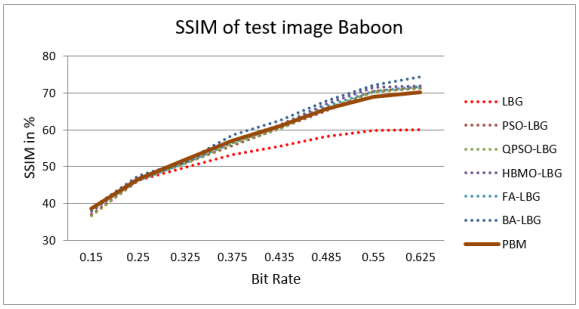

FIGURE 6. Baboon SSIM.

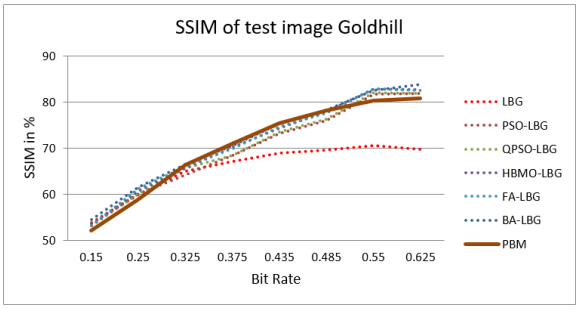

FIGURE 9. Gold Hill SSIM.

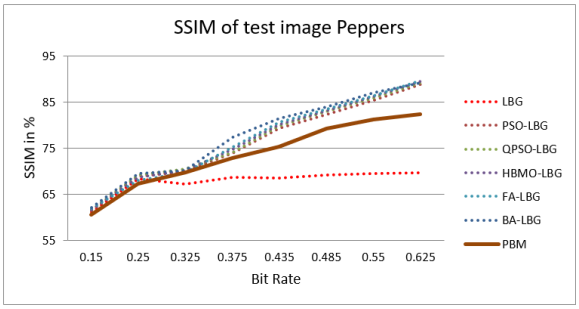

FIGURE 7. Peppers SSIM.

be in the range of [0-1], where 0 and 1 show no resemblance and high resemblance between the two images, respectively. The SSIM score (in percentage) of five test images using PBM is compared with the existing algorithms, as shown in Figs. 5, 6, 7, 8 and 9.

It can be observed from graph that the proposed algorithm attains high percentage of SSIM compared to LBG algorithm at the bit rates $\geq$ 0.375 for all test images and produces comparable results to LBG based algorithms.

The comparison of three reconstructed test images using LBG and LBG based algorithms using codebook of 64 and block size of 16 are shown in Figs. 12, 13, and 14. It is noted that the quality of the reconstruction image is superior to LBG and related algorithms.

A PSNR and iteration count comparison for test image Lena at the bitrates 0.325 and 0.55 are shown in Fig 10 and 11. It is noted that PBM algorithm consumes only 2 iterations whereas the other algorithms take 20 iterations to achieve approximately the same PSNR value.

As simulations are performed using different codebook sizes. Increasing the size of the codebook improves the quality of the resulting image but increases the number of comparisons between the codewords and training vectors, hence causes high computation time as well as provides a low compression ratio. Table (6) shows the average computation time of different test images using PBM and other related

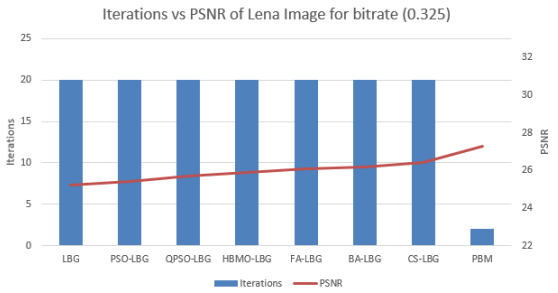

FIGURE 10. PSNR vs iterations at bitrate 0.325.

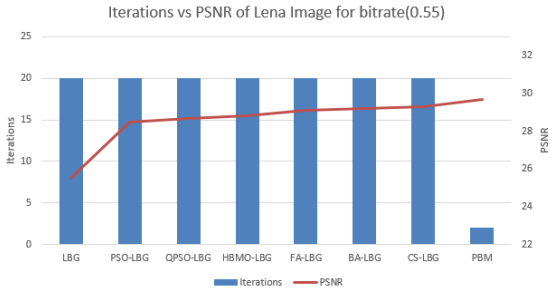

FIGURE 11. PSNR vs iterations at bitrate 0.55.

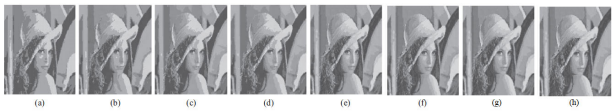

FIGURE 12. Reconstructed image of LENA using six algorithms: (a) LBG. (b) PSO-LBG. (c) QPSO-LBG. (d) HBMO-LBG. (e) FA-LBG. (f) BA-LBG. (g) CS-LBG. (h) PBM.

FIGURE 13. Reconstructed image of Barb using six algorithms: (a) LBG. (b) PSO-LBG. (c) QPSO-LBG. (d) HBMO-LBG. (e) FA-LBG. (f) BA-LBG. (g) CS-LBG. (h) PBM.

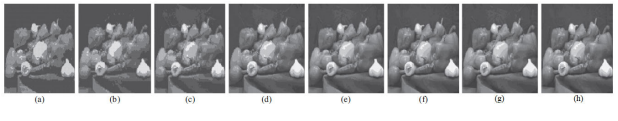

FIGURE 14. Reconstructed image of Peppers using six algorithms: (a) LBG. (b) PSO-LBG. (c) QPSO-LBG. (d) HBMO-LBG. (e) FA-LBG. (f) BA-LBG. (g) CS-LBG. (h) PBM.

algorithms. The computational time is calculated by performing the simulation five times for each test image.

LBG has less computational time for bit rate ≤ 0.375 but consumes high processing time for higher bit rates. It is noted that compared to LBG based algorithms such as PSO-LBG, QPSO-LBG, HBMO-LBG, FA-LBG, BA-LBG and CS-LBG has a high computational time compared to the proposed algorithm.

V. CONCLUSION

A histogram oriented PBM algorithm is proposed for the image compression. A comparable PSNR is achieved using peaks from the histogram by adding it to the predefined pattern mask to generate the codebook by carrying maximum repeated patterns. This method exploits to reduce the number of iterations of the algorithm hence decrease the processing time by reducing the number of closest match between the training vector and codewords of the codebook.

Based on simulation results, it is illustrated that the proposed algorithm is more superior to BA-LBG, FA-LBG, QPSO-LBG, HBMO-LBG, PSO-LBG and LBG in terms of low computational time at higher bitrates without compromising on PSNR and SSIM by consuming only two iterations.

The future researches may include the testing of algorithm for polychrome, monochrome and 3D images. Moreover, the results can be further improved if better edge oriented masks are added to the algorithm.

REFERENCES

[1] A. J. Hussain, A. Al-Fayadh, and N. Radi, "Image compression techniques: A survey in lossless and lossy algorithms," *Neurocomputing*, vol. 300, pp. 44–69, Jul. 2018.
[2] M. Wang, W. Xie, J. Zhang, and J. Qin, "Industrial applications of ultrahigh definition video coding with an optimized supersample adaptive offset framework," *IEEE Trans. Ind. Informat.*, vol. 16, no. 12, pp. 7613–7623, Dec. 2020.
[3] J. Xiong, H. Gao, M. Wang, H. Li, and W. Lin, "Occupancy map guided fast video-based dynamic point cloud coding," *IEEE Trans. Circuits Syst. Video Technol.*, early access, Mar. 2, 2021, doi: 10.1109/TCSVT.2021.3063501.
[4] R. C. Gonzalez and R. E. Woods, *Digital Image Processing*. London, U.K.: Pearson, 2018.
[5] R. M. Gray, "Vector quantization," *IEEE ASSP Mag.*, vol. 1, no. 2, pp. 4–29, Apr. 1984.
[6] H. Sun, K.-Y. Lam, S.-L. Chung, W. Dong, M. Gu, and J. Sun, "Efficient vector quantization using genetic algorithm," *Neural Comput. Appl.*, vol. 14, no. 3, pp. 203–211, Sep. 2005.
[7] C.-W. Tsai, S.-P. Tseng, C.-S. Yang, and M.-C. Chiang, "PREACO: A fast ant colony optimization for codebook generation," *Appl. Soft Comput.*, vol. 13, no. 6, pp. 3008–3020, Jun. 2013.
[8] R. Pizzolante, B. Carpentieri, and S. De Agostino, "Adaptive vector quantization for lossy compression of image sequences," *Algorithms*, vol. 10, no. 2, p. 51, May 2017.
[9] Y. Linde, A. Buzo, and R. M. Gray, "An algorithm for vector quantizer design," *IEEE Trans. Commun.*, vol. COM-28, no. 1, pp. 84–95, Jan. 1980.
[10] B. Huang and L. Xie, "An improved LBG algorithm for image vector quantization," in *Proc. 3rd Int. Conf. Comput. Sci. Inf. Technol.*, vol. 6, Jul. 2010, pp. 467–471.
[11] N. Rajpoot, A. Hussain, U. Ali, K. Saleem, and M. Qureshi, "A novel image coding algorithm using ant colony system vector quantization," in *Proc. Int. Workshop Syst., Signals, Image Process. (IWSSIP)*, Poznań, Poland, Sep. 2004.
[12] M. Kumar, R. Kapoor, and T. Goel, "Vector quantization based on self-adaptive particle swarm optimization," *Int. J. Nonlinear Sci*, vol. 9, no. 3, pp. 311–319, 2010.
[13] H.-M. Feng, C.-Y. Chen, and F. Ye, "Evolutionary fuzzy particle swarm optimization vector quantization learning scheme in image compression," *Expert Syst. Appl.*, vol. 32, no. 1, pp. 213–222, Jan. 2007.
[14] Y. Wang, X.-Y. Feng, Y.-X. Huang, D.-B. Pu, W.-G. Zhou, Y.-C. Liang, and C.-G. Zhou, "A novel quantum swarm evolutionary algorithm and its applications," *Neurocomputing*, vol. 70, nos. 4–6, pp. 633–640, 2007.
[15] M.-H. Horng and T.-W. Jiang, "Image vector quantization algorithm via honey bee mating optimization," *Expert Syst. Appl.*, vol. 38, no. 3, pp. 1382–1392, 2011.
[16] M.-H. Horng, "Vector quantization using the firefly algorithm for image compression," *Expert Syst. Appl.*, vol. 39, no. 1, pp. 1078–1091, 2012.

[17] X. Li, J. Ren, C. Zhao, T. Qiao, and S. Marshall, "Novel multivariate vector quantization for effective compression of hyperspectral imagery," *Opt. Commun.*, vol. 332, pp. 192–200, Dec. 2014.
[18] S. M. Hosseini and A.-R. Naghsh-Nilchi, "Medical ultrasound image compression using contextual vector quantization," *Comput. Biol. Med.*, vol. 42, pp. 743–750, Jul. 2012.
[19] D. Tsolakis, G. E. Tsekouras, and J. Tsimikas, "Fuzzy vector quantization for image compression based on competitive agglomeration and a novel codeword migration strategy," *Eng. Appl. Artif. Intell.*, vol. 25, no. 6, pp. 1212–1225, Sep. 2012.
[20] C. Karri and U. Jena, "Fast vector quantization using a bat algorithm for image compression," *Eng. Sci. Technol., Int. J.*, vol. 19, pp. 769–781, Jun. 2016.
[21] Y. Feng, Z. Lu, and H. Li, "Image coding based on classified vector quantisation using edge orientation patterns," *IET Image Process.*, vol. 11, no. 10, pp. 910–918, Oct. 2017.
[22] K. Chiranjeevi and U. R. Jena, "Image compression based on vector quantization using cuckoo search optimization technique," *Ain Shams Eng. J.*, vol. 9, no. 4, pp. 1417–1431, Dec. 2018.
[23] D. P. Rini, S. M. Shamsuddin, and S. S. Yuhaniz, "Particle swarm optimization: Technique, system and challenges," *Int. J. Comput. Appl.*, vol. 14, no. 1, pp. 19–26, 2011.
[24] G. T. C. Sekhar, R. K. Sahu, A. K. Baliarsingh, and S. Panda, "Load frequency control of power system under deregulated environment using optimal firefly algorithm," *Int. J. Electr. Power Energy Syst.*, vol. 74, pp. 195–211, Jan. 2016.
[25] S. Nag, "Vector quantization using the improved differential evolution algorithm for image compression," *Genet. Program. Evolvable Mach.*, vol. 20, no. 2, pp. 187–212, Jun. 2019.
[26] B.-H. Yuan and G.-H. Liu, "Image retrieval based on gradient-structures histogram," *Neural Comput. Appl.*, vol. 32, pp. 1–11, Jan. 2020.
[27] D. Cheng, V. Cammarota, Y. Fantaye, D. Marinucci, and A. Schwartzman, "Multiple testing of local maxima for detection of peaks on the (celestial) sphere," *Bernoulli*, vol. 26, no. 1, pp. 31–60, Feb. 2020.
[28] W. Khalaf, A. S. Mohammad, and D. Zaghar, "Chimera: A new efficient transform for high quality lossy image compression," *Symmetry*, vol. 12, no. 3, p. 378, Mar. 2020.
[29] Z. Tan, J. Song, X. Ma, S.-H. Tan, H. Chen, Y. Miao, Y. Wu, S. Ye, Y. Wang, D. Li, and K. Ma, "PCNN: Pattern-based fine-grained regular pruning towards optimizing CNN accelerators," 2020, *arXiv:2002.04997*. [Online]. Available: http://arxiv.org/abs/2002.04997
[30] M. L. P. Rani, G. S. Rao, and B. P. Rao, "An efficient codebook generation using firefly algorithm for optimum medical image compression," *J. Ambient Intell. Humanized Comput.*, vol. 12, pp. 1–13, Feb. 2020.
[31] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: From error visibility to structural similarity," *IEEE Trans. Image Process.*, vol. 13, no. 4, pp. 600–612, Apr. 2004.

MUHAMMAD BILAL received the B.Sc. degree (Hons.) in computer system engineering and the M.S. degree in computer system engineering from the University of Engineering and Technology, Peshawar, Pakistan, in 2007 and 2013, respectively. He is currently a Ph.D. Scholar with the Department of Electrical Engineering, CECOS University of IT and Emerging Sciences, Peshawar. He has authored journal articles and conference papers. His research interests include image processing and communication technologies.

ZAHID ULLAH (Member, IEEE) received the B.Sc. degree (Hons.) in computer system engineering from the University of Engineering and Technology, Peshawar, Pakistan, in 2006, the M.S. degree in electronic, electrical, control, and instrumentation engineering from Hanyang University, South Korea, in 2010, and the Ph.D. degree in electronic engineering from the City University of Hong Kong, in 2014. He has previously worked as an Associate Professor and the Head of the Department of Electrical Engineering, CECOS University of IT and Emerging Sciences, Peshawar. He is currently an Assistant Professor and the Head of the Department of Electrical and Computer Engineering, Pak-Austria Fachhochschule: Institute of Applied Sciences and Technology, Haripur, Pakistan. He has authored prestigious journal articles, conference papers, and holds patents in his name in the field of FPGA-based TCAM. His research interests include low-power/high-speed CAM design on FPGA, low-power/high-speed VLSI design, and embedded systems.

IHTESHAM UL ISLAM received the B.S. degree in computer systems engineering from the University of Engineering and Technology, Peshawar, Pakistan, in 2006, the M.S. degree in electronics and communication engineering from Myongji University, South Korea, in 2009, and the Ph.D. degree in computer and control engineering from the University of Politecnico di Torino, Italy, in 2015. From 2009 to 2012, he was working as a Lecturer with the FAST-National University of Computer and Emerging Sciences, Peshawar. From 2015 to 2020, he was working as an Assistant Professor with the Department of Computer Science and IT, Sarhad University, Peshawar. He is currently working as an Associate Professor with the Department of Computer Software Engineering, Military College of Signals, National University of Sciences and Technology, Pakistan. His research interests include computer vision and machine learning.