Ingénierie des Systèmes d'Information
Vol. 31, No. 8, August, 2026, pp. 2653-2663
Journal homepage: http://iieta.org/journals/isi

Hybrid Lossy Image Compression Using Bilateral Filtering, Vector Quantization, and Entropy Coding

Ibtihal Hussein Kadhim, Ahmed Fanfakh*, Esraa Hadi Alwan

Department of Computer Science, College of Science for Women, University of Babylon, Hillah 51002, Iraq

Corresponding Author Email: ahmed.fanfakh@uobabylon.edu.iq

https://doi.org/10.18280/isi.310812

Received: 6 May 2026
Revised: 10 July 2026
Accepted: 23 July 2026
Available online: 31 August 2026

Keywords:
lossy image compression, vector quantization, Linde–Buzo–Gray algorithm, bilateral filtering, delta coding, Huffman coding

ABSTRACT

Efficient image compression requires an appropriate balance among compression ratio, reconstruction quality, and computational cost. This study proposes a hybrid lossy image compression framework that combines bilateral filtering, Linde–Buzo–Gray (LBG)-based vector quantization (VQ), delta coding, and Huffman entropy coding. Bilateral filter was first applied as an edge-preserving preprocessing step to reduce local intensity variations while retaining important structural information. The preprocessed image is then partitioned into non-overlapping blocks and encoded using LBG-based VQ. To further reduce redundancy in the resulting index sequence, delta coding was employed before Huffman coding was applied to generate the final compressed bitstream. The framework was evaluated on four standard 512 × 512, 8-bit grayscale benchmark images using compression ratio (CR), peak signal-to-noise ratio (PSNR), mean squared error (MSE), structural similarity index measure (SSIM), and execution time. Across the test images, the proposed method achieved an average CR of 26.09:1, an average PSNR of 29.88 dB, an average SSIM of 0.8447, and an average MSE of 67.04. The average LBG execution time was approximately 0.3104 s. Additional experiments on block and codebook sizes confirmed that reconstruction quality, compression efficiency, and computational cost vary systematically with parameter selection. Overall, the proposed framework provides a computationally practical approach to lossy grayscale image compression while maintaining acceptable reconstruction quality at relatively high compression ratios.

1. INTRODUCTION

Image compression plays an important role in modern digital imaging systems by reducing the data required for storage and transmission. As digital images are increasingly used in medical imaging, remote sensing, multimedia communications, and signal processing, the need for efficient compression methods continues to grow. Digital images often contain spatial and statistical redundancy that can be exploited to represent them using less data [1-3]. This is particularly important in applications such as digital pathology, where large collections of image data must be stored, transmitted, and retrieved efficiently [4, 5]. Image compression techniques are generally classified into two main categories: lossless and lossy compression [6-9]. Among widely used lossy compression techniques, transform-based methods such as the Discrete Cosine Transform (DCT), Discrete Wavelet Transform (DWT), and Singular Value Decomposition (SVD) reduce redundant information through different mathematical representations of the data [10].

Some learning-based image compression methods, such as learned vector quantization (VQ) and rate-adaptive autoencoder architectures, have demonstrated significant improvements in both reconstructed-image quality and rate-distortion performance. Although these methods are very capable, they typically utilize intricate network architectures, require long training times, and are resource-intensive. These factors could potentially prohibit the use of these methods in either resource-limited environments or time-sensitive compression situations [11-13].

This study focuses on improving the performance of conventional image-compression methods through a hybrid lossy compression framework. The framework comprises two main stages: image preprocessing and compression. In the first stage, bilateral filtering is applied to the input image to reduce local intensity variations while preserving edges. In the second stage, LBG-based vector quantization performs lossy data reduction. Delta coding is subsequently applied to exploit the local correlation between consecutive quantization indices, followed by Huffman coding to reduce the statistical redundancy of the resulting sequence [14]. Compared with conventional LBG-based image-compression methods, the proposed framework integrates edge-preserving bilateral filtering, vector quantization, delta coding, and Huffman coding within a unified pipeline to improve compression efficiency while maintaining reconstruction quality. The contribution of this work is summarized as follows:

A hybrid lossy image compression framework is proposed by integrating bilateral filtering, LBG-based VQ, delta coding, and Huffman coding into a unified compression pipeline. An edge-preserving bilateral filtering stage is incorporated before VQ to improve block homogeneity, thereby reducing

quantization distortion and improving reconstruction quality. Delta coding is employed to exploit the correlation between consecutive VQ indices before Huffman coding, leading to improved entropy coding efficiency by reducing coding redundancy. The proposed framework is experimentally evaluated using standard benchmark images and demonstrated a balance between CR, reconstruction quality, and computational cost.

The remainder of this paper is organized as follows: Section 2 discussed related work, and Section 3 reviews the theoretical background. Section 4 describes the proposed compression method. Section 5 presents the experimental results and discussion. Finally, Section 6 concludes the paper.

2. RELATED WORK

The necessity for compression techniques that reduce storage and transmission requirements while maintaining acceptable reconstruction quality has increased due to the growing volume of digital image data. The underlying principles, computational demands, compression performance, and applicability of various lossy image compression techniques vary [15].

Conventional transform-based and vector-quantization-based methods, as well as learning-based strategies, have been used in recent image compression research. While learning-based approaches use data-driven models to generate compact image representations [11, 13], traditional approaches primarily rely on signal processing, mathematical transformations, quantization, and entropy coding. Representative approaches from these perspectives are reviewed in the discussion that follows, with special attention paid to computational needs, compression performance, and reconstruction quality.

2.1 Traditional lossy compression techniques

To enhance reconstructed image quality and minimize compression artifacts, Kulkarni and Dixit [1] devised a hybrid image compression technique that incorporates the DCT, SVD, and adaptive variable quantization. A medical image compression technique based on the DWT, a reduction operation, and Huffman coding was proposed by Thomas et al. [6]. A hybrid image compression technique based on DWT and SVD was introduced by Vijaywargiya and Pandey [2], with performance mainly assessed in terms of peak signal-to-noise ratio (PSNR) and CR. A hybrid lossy compression technique combining principal component analysis (PCA), DWT, and canonical Huffman coding (CHC) was devised by Ranjan and Kumar [16]. The technique was assessed using both grayscale and color images.

These studies demonstrate that transform-based techniques can improve compression performance through various combinations of transformations and entropy coding. However, they primarily focus on improving transform domain processing and do not report execution time measurements.

2.2 Vector-quantization-based compression techniques

Enhancing the efficiency of vector-quantization-based image compression has been the topic of several studies. The Fast Linde–Buzo–Gray (FLBG) approach was proposed by Bilal et al. [17]. It uses rescaling based on bilinear interpolation to reduce computational cost by minimizing the number of comparisons between the training vectors and the codebook. An improved sine-cosine algorithm (ISCA) was suggested by Ghadami and Rahebi [18]. To minimize compression error and maximize codebook generation in LBG-based VQ. Abedi and Al-Baghdadi [3] formulated VQ as a multi-objective optimization problem using a genetic algorithm. While Camacho-Gonzalez et al. [19] provided a more recent VQ-based multi-objective evolutionary framework that jointly analyzes image quality and compression level, Rahebi [20] used the Whale Optimization Algorithm (WOA) for VQ codebook optimization. These studies show that codebook optimization and design continue to be crucial factors in enhancing VQ-based image compression performance. However, the integration of preprocessing, entropy coding within a unified compression framework remains limited.

Table 1. Comparison of representative image compression methods

Reference	Method	Category	CR	PSNR	Time
[1]	Adaptive DCT-SVD Hybrid	TLC	24.37 (Avg.)	34.69 dB (Avg.)	-
[2]	DWT-SVD Hybrid	TLC	1.85–2.31	86.8–89.0 dB	-
[6]	DWT-Huffman coding	TLC	1.61	54.66 dB	-
[11]	Adaptive LVQ (Mult irate LVQ)	LLC	32	28–38 dB	-
[12]	LVQAC (Adaptive Combining LVQ)	LLC	32	28–38 dB	-
[13]	QARV (Quantization-Aware ResNet VAE)	LLC	-	32.47 dB	-
[16]	2D DWT + PCA + Canonical Huffman (CHC)	TLC	14.3	28–38 dB	96.59 s
[17]	FLBG (Fast LBG with Pre-Scaling)	TLC	-	24–26 dB	4.02 s
[18]	Sine–Cosine Optimized VQ (SCA-VQ / ISCA-LBG)	TLC	12.8–53.3	27.24 dB	-
[19]	RAQ-VAE	LLC	8–80	≈28–36 dB	-
[20]	WOA-Optimized VQ (Whale Optimization VQ)	LLC	-	32.85 dB	-

Note: CR = compression ratio; PSNR = peak signal-to-noise ratio; TLC = traditional lossy compression; LLC = learning-based lossy compression; DCT = discrete cosine transform; SVD = singular value decomposition; DWT = discrete wavelet transform; VQ = vector quantization; LVQ = lattice vector quantization; LVQAC = lattice vector quantization with adaptive companding; QARV = Quantization-Aware ResNet variational autoencoder; VAE = variational autoencoder; PCA = principal component analysis; CHC = canonical Huffman coding; LBG = Linde–Buzo–Gray; FLBG = Fast Linde–Buzo–Gray; SCA-VQ = sine cosine algorithm-based vector quantization; ISCA-LBG = Improved Sine Cosine Algorithm-based Linde–Buzo–Gray; RAQ-VAE = rate-adaptive quantization variational autoencoder; WOA = Whale Optimization Algorithm.

2.3 Learning-based lossy compression techniques

Strong rate-distortion performance has been demonstrated by recent developments in learning-based image compression, especially when adaptive quantization techniques are incorporated. A multirate neural image compression system based on adaptive lattice VQ was presented by Xu et al. [11], allowing for compression at several rates. Zhang and Wu [12]

proposed LVQAC for efficient learned image compression, which combines lattice VQ with spatially adaptive companding. QARV, a quantization-aware ResNet variational autoencoder for variable - rate lossy image compression, was proposed by Duan et al. [13]. These studies demonstrate how adaptive quantization is becoming increasingly important in learning picture compression frameworks.

In Table 1, representative image compression techniques chosen to demonstrate several strategies pertinent to the current investigation are compiled. Because of their connection to the compression stages used in the suggested framework, hybrid and VQ-based techniques received special attention, and new learning-based techniques were added to place the study within the existing body of research on image compression. When available, the comparison takes into account the performance metrics provided by the individual studies, such as PSNR, CR, and execution time. Although these methods provide strong compression performance, they rely on complex neural network architectures and computationally intensive training, which may limit their applicability in resource-constrained environments.

Despite these advances, limited attention has been given to hybrid VQ-based frameworks that simultaneously improve compression efficiency while maintaining reconstruction quality with low computational cost. This research gap motivated the proposed framework, which integrates bilateral filtering, VQ, Delta coding, and Huffman coding.

3. BACKGROUND

3.1 Bilateral filtering

Bilateral filtering is a non-linear edge-preserving smoothing technique used to reduce noise in images while maintaining significant image structures. Unlike traditional linear filters such as Gaussian filtering, it incorporates both spatial proximity and intensity similarity when computing pixel weights [21]. The filtering operation is defined as:

$$I_{\mathrm{BF}}(x) = \frac{1}{W_x}\sum_{y\in\Omega} I(y) \; exp\left(-\frac{\|x-y\|^2}{2\sigma^2}\right) exp\left(-\frac{(I(x)-I(y))^2}{2\sigma^2}\right) \quad (1)$$

$$W_x = \sum_{y\in\Omega} \exp\left(-\frac{\|x-y\|^2}{2\sigma^2}\right)\exp\left(-\frac{(I(x)-I(y))^2}{2\sigma^2}\right) \quad (2)$$

$I_{BF}(x)$ is the value of the filtered pixel at position x, $I(y)$ is the intensity of the adjacent pixel in the window Ω, W_x is the normalization factor, σ_s controls the effect of spatial distance, σ_r controls intensity similarity, $\|x-y\|$ is the spatial distance, and $|I(x)-I(y)|$ is the intensity difference. The Bilateral filter was chosen for its ability to reduce local variations in pixel intensity while preserving important edge information [22, 23]. Unlike Gaussian filtering, which smooths both noise and edges, and median filtering, which is primarily effective for impulse noise, bilateral filtering preserves edge information while reducing local intensity variations, making it more suitable as a preprocessing stage for VQ. The filter's behavior is primarily controlled by the neighborhood window diameter (d), the intensity similarity parameter (σ_{Color}), and the spatial distance parameter (σ_{Space}). Generally, increasing these parameters results in stronger or wider spatial smoothing, while smaller values limit the smoothing effect. During initial experiments, several preprocessing settings were tested, and it was observed that stronger smoothing led to a noticeable loss of image detail and reconstruction quality. Therefore, (d = 5), (σ_{Color} = 20), and (σ_{Space} = 20) were adopted, with conservative blending of the original and filtered images according to the ratio ($0.8I + 0.2I_{BF}$), to limit excessive smoothing before VQ. The effect of this setup is experimentally evaluated in Section 5.4.

3.2 Vector quantization

VQ is a lossy coding technique that maps each multidimensional input vector to the nearest codeword in a finite codebook. The corresponding codeword index is stored instead of the complete input vector, thereby reducing the amount of data required for image representation [17]. Because input vectors are replaced by representative codewords, codebook design directly affects quantization distortion and reconstruction quality [3, 18].

3.3 Linde–Buzo–Gray algorithm and computational complexity

The LBG algorithm is widely used for codebook generation in VQ systems due to its distortion-minimization capability. However, the iterative distance calculations and nested loops involved in training and encoding result in high computational complexity and long execution times, especially for large images or large codebooks [17, 20]. The average distortion between input vectors and codewords is defined as:

$$D = \frac{1}{N_b}\sum_{i=1}^{N_b}\left\|X_i - C_j(i)\right\|^2 \quad (3)$$

where, ($C_j(i)$) denotes the nearest codeword assigned to the input vector (X_i), and (N_b) denotes the total number of vectors (image blocks).

3.4 Delta coding for redundancy reduction

Delta coding (or differential coding) is a lossless technique that exploits the strong local correlation between adjacent VQ indices. Instead of encoding absolute values, the difference between successive indices $\Delta_i = X_i - X_{i-1}$ is used. This can reduce the entropy of the index sequence, improving the efficiency of subsequent entropy coding stages without introducing additional reconstruction distortion [24].

3.5 Huffman coding

Huffman coding is a common lossless entropy coding method used in image compression. It is commonly applied after quantization stages to lower the bit rate even more. In VQ-based systems, the image is represented by codebook indices that may contain statistical redundancy because some index values accrue repeatedly. Huffman coding exploits this redundancy by assigning shorter codewords to frequently occurring symbols and longer codewords to less frequent symbols. This improves compression efficiency without introducing additional reconstruction distortion [24, 25].

$$I_x(i) = -Log_2 P(x_i) \quad (4)$$

$$H_{(X)} = -\sum_{i=1}^{n} P(x_i)\, Log_2 P(x_i) \quad (5)$$

where, $I_x(i)$ is the information content of symbol x_i, $P(x_i)$ is the probability of occurrence of that symbol, n is the number of distance symbol, and $H(X)$ is the source entropy of bits per symbol. These equations define the information content of each symbol and the average information (entropy) of the source, which forms the theoretical basis of Huffman coding. Huffman coding assigns variable-length codes to symbols according to their probabilities to approach this entropy limit.

4. PROPOSED LOSSY COMPRESSION METHOD

A hybrid framework for compressing grayscale images is proposed to improve compression efficiency while keeping the quality of the reconstruction and the cost of the calculations reasonable. The method uses a structured pipeline to combine several traditional techniques to exploit both spatial and statistical redundancy in the image data.

Following that step, delta coding is applied to the generated index sequence to represent the differences between consecutive values and reduce redundancy from the data stream.

Figure 1 illustrates the four main stages of the proposed framework. Bilateral filtering performs edge-preserving noise reduction before compression. LBG-based VQ performs the primary lossy compression by representing image blocks with codebook indices. Delta coding exploits the correlation between successive indices to reduce statistical redundancy, while Huffman coding performs the final lossless entropy coding to further reduce the compressed bitstream size.

Finally, the method applies Huffman coding as the entropy-coding stage to compress the data further and provide the final compressed version of the data stored in a file according to Algorithm 1, which describes the main steps in the proposed compression framework.

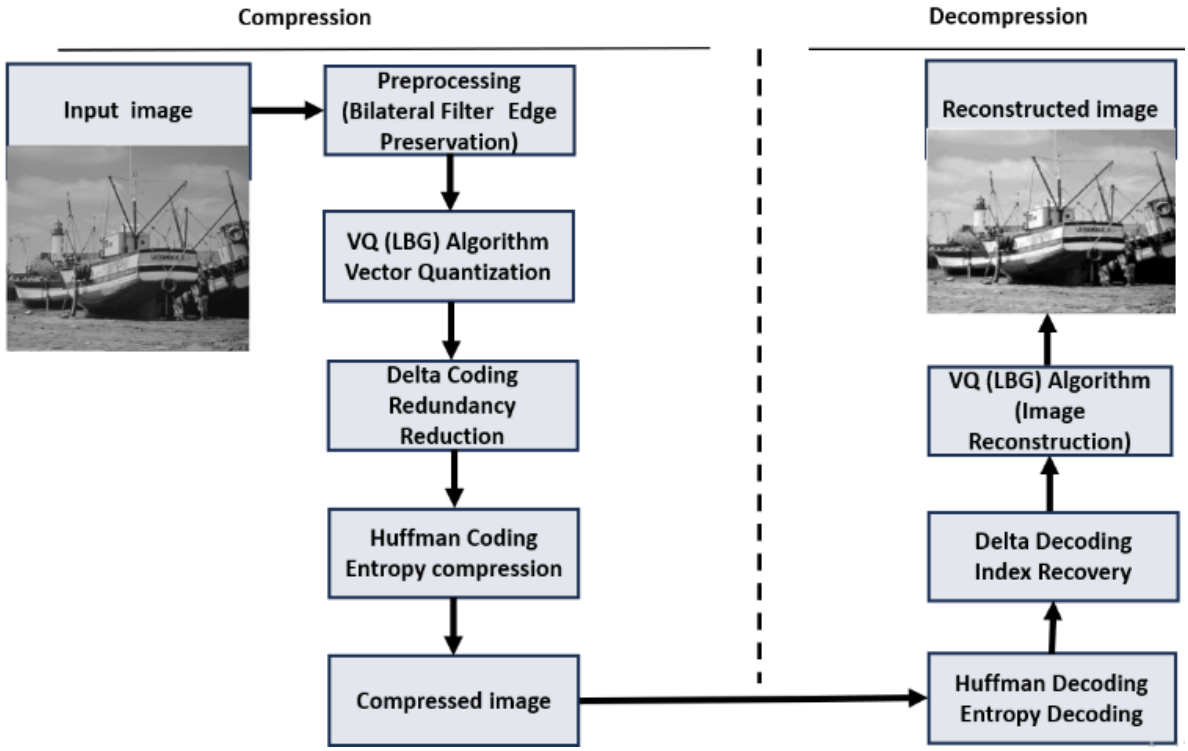

Figure 1. Block diagram of the proposed image compression framework

Algorithm 1. Proposed Compression Algorithm (Vector Quantization + Delta + Huffman)

Input:
in: Grayscale image *I* with size W × H
B: Block size
K: **C**odebook size
δ: Splitting factor
ε: Convergence threshold
Iter_max: Maximum Lloyd iterations

Output:
Pack: Compressed image package
{*C*, *H*, *BS*, *W*, *h*, *B*, *K*}

where, *C*: codebook, *H*: Huffman header, *BS*: compressed bitstream, Wand h are width and height, respectively B is the block size K is codebook size.

Start
1: I_p ← Bilateral_Filter_Preprocessing (*I*)
2: (*C*, *L*) ← Vector_Quantization_LBG (I_p, *B*, *K*, *δ*, *ε*, Iter_max) // **Algorithm 3**
3: *D* ← Delta_Encode (*L*, offset = 128)
4: (*H*, *BS*) ← Huffman_Encode (*D*, Max Symbol = 255)
5: Pack ← {*C*, *H*, *BS*, *W*, *h*, *B*, *K*}
End

All parameters are required for decoding and exact reconstruction of the original image. The proposed method combines preprocessing, VQ, delta encoding, and Huffman coding into one system. From noise suppression to entropy coding, each step cuts down on redundancy, which leads to efficient compression and a small image representation. The following sections go into great detail about each step of the proposed method.

4.1 Preprocessing (bilateral filtering)

Several preprocessing settings were tested, but the selected setting was chosen because it achieved a better balance between compression efficiency and detail preservation, while some other settings resulted in a noticeable reduction in PSNR. In the first stage, which applies a bilateral filter is applied to reduce noise while preserving edge information, as described in Algorithm 2.

Algorithm 2. Bilateral Filter Preprocessing Algorithm

Input:
I: Grayscale image of size W × H
$\boldsymbol{\Sigma s}$, $\boldsymbol{\sigma r}$: Spatial and range parameters
Output:
$\boldsymbol{I}_{\text{final}}$: preprocessed image
Start
1: For each pixel $\boldsymbol{x}$ in $\boldsymbol{I}$ do
2: Define a spatial neighborhood Ω around x
3: Compute spatial weights $\boldsymbol{w_s(x, y)}$ for all $y \in \Omega$
4: Compute range weights $\boldsymbol{w_r(x, y)}$ based on intensity differences
5: Compute bilateral weights $\boldsymbol{w(x, y) = w_s(x, y) \times w_r(x, y)}$
6: Normalize weights by dividing by their sum
7: Compute filtered value $\boldsymbol{I_{BF}(x)}$ as weighted average of neighbors
8: End For
9: $\boldsymbol{I}_{\text{final}} \leftarrow \boldsymbol{I_{BF}}$
End

4.2 Linde–Buzo–Gray algorithm

In Algorithm 3, the VQ operation on an image block is represented by the index of the nearest codeword in a trained codebook. In this study, the preprocessed 512 × 512 grayscale image was divided into non-overlapping 4 × 4 blocks. Each block is then reshaped into a 16-dimensional vector.

Algorithm 3. Vector Quantization (Linde–Buzo–Gray)

Input:
$\boldsymbol{I} \in \mathbb{R}^{W \times H}$ // grayscale image
$\boldsymbol{B}$ // block size (e.g., 4 × 4)
$\boldsymbol{K}$ // number of codewords
$\boldsymbol{\delta}$ // small perturbation (splitting factor)
$\boldsymbol{\varepsilon}$ // convergence threshold
Iter_max // maximum iterations
Output:
$\boldsymbol{C} = \{\boldsymbol{C_1}, \boldsymbol{C_2}, \ldots, \boldsymbol{C_K}\}$ // codebook
$\boldsymbol{L} = \{\boldsymbol{L_1}, \boldsymbol{L_2}, \ldots, \boldsymbol{L_N}\}$ // label for each block
Start
1. Divide the image into $B \times B$ blocks and vectorize each block: $\boldsymbol{x_i} \in \mathbb{R}^{\boldsymbol{d}}$, $\boldsymbol{d = B^2}$
2. Form the training set $\boldsymbol{T} = \{\boldsymbol{x_1}, \boldsymbol{x_1}, \ldots, \boldsymbol{x_N}\}$
3. Initialize codebook with one codeword: $C_1 = \frac{1}{N}\sum_{i=1}^{N} x_i$
4. $k \leftarrow 1$
5. **While** $k < K$ **do**
// **Step 1**: Split codewords to double the codebook size
for $c = 1$ to k:
$C_c \leftarrow C_c \times (1 + \delta)$
$C_{\{k+c\}} \leftarrow C_c \times (1 - \delta)$
$k \leftarrow 2 \times k$
// **Step 2:** Refine codewords using Lloyd iteration
repeat
// Assign each vector to the nearest codeword
For $i = 1$ to N:
L [i] = $\text{argmin}_j \| x_i - C_j \|^2$ // i.e., choose the symbol closest to the block i
// Update each codeword as the mean of assigned vectors
for $j = 1$ to k:
if $n_j > 0$:
$C_j = (1/n_j) \Sigma\{x_i$ assigned to $C_j\} x_i$
else:
optionally reinitialize C_j
// Compute average distortion
$D = (1/\text{N}) \Sigma_i \| x_i - C_{Li} \|^2$
$\Delta D = |D_{\text{prev}} - D| / (D + 1\text{e}^{-9})$
$D_{\text{Prev}} = D$
until $\Delta D < \varepsilon$ or iterations ≥ Iter_max
End While
Return C and L

4.2.1 Linde–Buzo–Gray codebook training

The LBG algorithm was used to generate the codebook (K = 64) because it effectively reduces quantization distortion. Training began with a single codeword initialized that is set to the global mean of all vectors. The codebook was expanded iteratively by splitting each codeword using a small perturbation ($\delta = 0.01$):

$$cw^{+} = cw(1 + \delta), cw^{-} = cw(1 - \delta) \tag{6}$$

Lloyd's algorithm assigns each training vector its nearest codeword and updates the centroids after each split. The iteration stopped when the improvement in distortion fell below the convergence threshold or when the maximum of 100 iterations was reached. In most cases, the algorithm terminates before then, reducing execution time.

4.2.2 Vector quantization encoding (label generation)

The index of the closest codeword is used to encode each 4 × 4 block, which creates a label stream that is N = 16,384 indices. An 8-bit integer is used to store each label. This stream forms the compressed image representation at the VQ stage. Subsequently, delta coding and Huffman coding are used to process it further for lossless compression.

4.2.3 Vector quantization decoding (reconstruction)

During decompression, each index is replaced by its corresponding codeword from the codebook. The 4 × 4 block is returned to its original position. To produce the reconstructed image, pixel values are rounded to 8-bit integers.

Figure 2 illustrates the LBG codebook generation process, showing iterative splitting of codewords and centroid updates until the codebook size is reached [17].

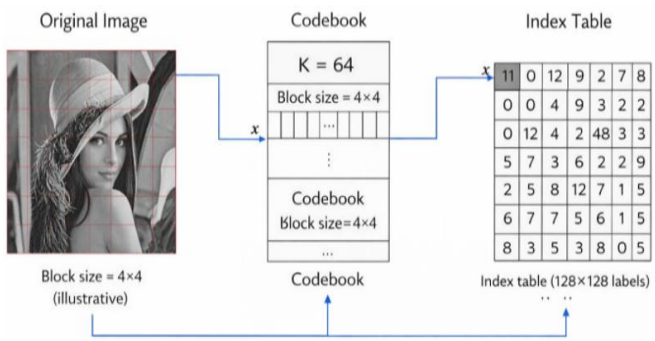

Figure 2. Linde–Buzo–Gray (LBG) codebook generation process

4.3 Delta coding (differential coding)

As described in Algorithm 4, delta coding is a simple form of lossless compression that encodes data as the differences

between successive VQ indices rather than the original index values themselves. Because successive VQ indices in most images have a high degree of correlation, delta coding can provide improve compression efficiency when entropy coding techniques are applied to delta-coded VQ indices. The first index is saved using the absolute value, whereas each subsequent index is encoded using its difference from the preceding index.

$$\Delta_i = L_i - L_{i-1} \qquad i = 0, \ldots, N \tag{7}$$

where, L_i and $L_{i\text{-}1}$ denote the current and precedent VQ indices, respectively, and Δ_i denotes the difference between them.

Algorithm 4. Delta Encoding Algorithm

Input:
$L \in \{0, \ldots, K-1\}^N$ // label array of length N
offset $\in \mathbb{N}$ // constant shift (e.g., 128)
Output:
$D \in \{0, \ldots, 255\}^N$ // unsigned delta stream
1. $D[0] \leftarrow L[0]$
2. for $i = 1$ to $N - 1$ do
$D[i] \leftarrow (L[i] - L[i-1]) + \text{offset}$
end for
Return *D*

This process can increase the frequency of recurring values in the delta-coded index sequence, thereby improving the efficiency of subsequent Huffman coding.

4.4 Huffman coding

As described in Algorithm 5, Huffman coding was applied as the final compression stage to the delta-coded index sequence. A Huffman tree was constructed from the symbol frequencies with shorter codewords assigned to more frequent symbols. The resulting coding table from the encoding and decoding process supported to accelerate conversion between symbols and codewords by minimizing the overall size of the header, by including only essential decompression information.

Algorithm 5. Huffman Encoding of Delta-coded VQ indices

Input:
D // delta-coded index sequence
Output:
H*, *BS // header and compressed bitstream.

1. Compute symbol frequencies from *D*.
2. Build Huffman tree using a min-heap.
3. Generate variable-length codes for each symbol.
4. Encode *D* into bitstream *BS* using assigned codes.
5. Store required decoding information in header *H*.
6. Return *H* and *BS*.
End

This framework integrates VQ, delta coding, and Huffman coding into a cohesive compression pipeline, leveraging both spatial and statistical redundancy within the image.

As described in Algorithm 6, the decompression process works by reversing the compression stages to produce a reconstructed image.

This process recovers the image by sequentially decoding the Huffman symbols, reversing the delta code, and reconstructing the image using the VQ codebook.

Algorithm 6. Proposed Image Decompression Algorithm

Input:
Pack: {*C*, *H*, *BS*, *W*, *h*, *B*, *K*}, offset = 128
N: Length(*D*)
Output:
I_{rec} (reconstructed image)
Start
1: D ← Huffman _Decode (*H*, *BS*)
2: *L* [0] ← *D* [0]
3: for *i* = 1 to *N* − 1 do
L[*i*] ← (*D*[*i*] − offset) + *L*[*i* − 1]
end for
4: for *i* = 0 to *N* -1do
$x_i \leftarrow C[L[i]]$
end for
5: I_{rec} ← reshape _blocks (x_i, *B*, *W*, *h*)
6: return I_{rec}
End

4.5 Evaluation criteria

To evaluate the performance of the proposed image compression method, commonly used quantitative metrics were employed to assess compression efficiency, reconstruction quality, and computational cost [17, 19, 26, 27].

4.5.1 Compression ratio

The CR measures the reduction in data size achieved after compression and is defined as:

$$CR = \frac{B_{\text{original}}}{B_{compression}} \tag{8}$$

where, $B_{\text{compression}}$ and B_{original} indicate the dimensions of the original and compressed images, respectively.

4.5.2 Mean squared error

Mean squared error (MSE) quantifies the average distortion between the original and reconstructed images:

$$MSE = \frac{1}{M * N} \sum_{i=0}^{M-1} \sum_{j=0}^{N-1} [I(i,j) - \text{Irec}(i,j)]^2 \tag{9}$$

where

M: Number of rows (image height)
N: The number of columns (image width)
I(*i*, *j*): Pixel intensity at row *i* and column *j* in the original image
$\bar{I}(i,j)$: The reconstructed pixel value at row *i* and column *j*
Lower MSE values indicate better reconstruction accuracy.

4.5.3 Peak signal-to-noise ratio

PSNR evaluates the quality of the reconstructed image based on the MSE and is calculated:

$$PSNR = 10log10(\frac{{Max_i}^2}{MSE}) \tag{10}$$

where, $MAX_i = 255$ for 8-bit grayscale images. Higher PSNR values indicate better reconstruction quality.

4.5.4 Structural similarity index measure

Structural similarity index measure (SSIM) is used to evaluate the similarity between the original image and the reconstructed image based on brightness, contrast, and structural information, where the ideal value is 1, indicating perfect similarity [27].

$$SSIM(x,y) = \frac{(2\mu_x\mu_y + C_1)(2\sigma_{xy} + C_2)}{(\mu_x^2 + \mu_y^2 + C_1)(\sigma_x^2 + \sigma_y^2 + C_2)} \quad (11)$$

where,

μ_x, μ_y: Mean intensity of images x and y

σ_x^2, σ_y^2: Variance of x and y

σ_{xy}: Covariance between x and y

C_1, C_2: Small constants to stabilize the division

5. EXPERIMENTS AND RESULTS

This section presents the experimental evaluation of the proposed image compression framework. The method's performance was assessed in terms of compression efficiency, reconstruction quality, and execution time using standard grayscale test images.

5.1 Experimental setup

Four standard grayscale Bitmap (BMP) images were used for all experiments (Lena, Cameraman, Peppers, and Boat), all of size 512 × 512 pixels, 8-bit depth. To evaluate the proposed compression framework, reference images were selected to provide a variety of spatial and structural features. The "Lena" image combines smooth areas, edges, and fine details. The "Cameraman" image features well-defined object boundaries and relatively homogeneous regions, while the "Peppers" image contains curved structures and gradual intensity variations. The "Boat" image contains more complex textures and edge patterns. These images enabled the framework to be evaluated using different image characteristics while maintaining the same resolution and compression settings. Each image was divided into non-overlapping 4 × 4 blocks, and a VQ codebook containing K = 64 codewords was generated for each image.

The experiments were conducted on a Linux-based operating system. The experimental platform was equipped with a 13th Generation Intel® Core™ i7-13650HX processor (2.60 GHz), featuring 14 physical cores (6 Performance cores and 8 Efficient cores), 20 logical threads, and 24 GB RAM.

Preprocessing, VQ, delta coding, Huffman coding, and the remaining components of the proposed framework were implemented in C and compiled using GCC with the -O2 optimization level. The framework was initially developed using Python in Google Colab, whereas the final experiments and performance measurements were conducted using the C implementation on Linux.

5.2 Results and discussion

The proposed compression framework was implemented and tested using a set of standard images. Figure 3 illustrates the original 512 × 512 grayscale BMP test images used to evaluate the proposed framework. These standard images are commonly used in image compression studies to facilitate comparison and support the reproducibility of the experimental results.

Figure 3. Original 512 × 512 grayscale BMP test images used in the experiments (Lena, Cameraman, Peppers, and Boat)

Table 2. The performance of the proposed framework on four standard grayscale images

Image	CR:1	PSNR (dB)	MSE	SSIM	Time(s)
Lena	23.45	30.02	64.75	0.8384	0.3368
Cameraman	29.52	30.08	63.84	0.8908	0.3088
Peppers	24.66	29.88	66.89	0.7967	0.2877
Boat	26.71	29.52	72.66	0.8528	0.3083

Note: CR = compression ratio, PSNR = peak signal-to-noise ratio, MSE = mean squared error, SSIM = structural similarity index measure.

Table 2 presents the experimental results of the proposed framework on standard grayscale test images, including CR, PSNR, MSE, SSIM, and execution time. It also illustrates how performance differed among the four test images due to variations in their structural features and visual content. The distribution of training vectors and the resulting VQ indices are influenced by variations in texture, edges, and local intensity patterns. This variation effect codebook representation and the effectiveness of subsequent entropy coding. As a result, variations in reconstruction quality and CR were observed among the evaluated images. Instead of representing the performance of a single representative image, the values presented in the abstract provide averages across the four test images.

Table 3. Stage-wise performance comparison of the proposed method

Method Stage	CR(:1)	PSNR(dB)	Time(s)
LBG	15.06	29.56	0.3383
LBG + Huffman	22.04	29.56	0.3904
LBG + Delta + Huffman	23.01	29.56	0.3908
Filter + LBG + Delta + Huffman	23.45	30.02	0.4428

Note: CR = compression ratio, PSNR = peak signal-to-noise ratio, LBG = Linde–Buzo–Gray.

5.2.1 Stage-wise performance analysis

This subsection presents a stage-wise analysis of the

proposed framework using the Lena image as a representative benchmark. Table 3 shows the contribution of each stage to compression efficiency, reconstruction quality, and execution time.

The findings demonstrate that there is not enough redundancy in image data to take advantage of compression by using a single compression stage. LBG VQ alone on a 512 × 512 grayscale version of the Lena image with 64 entries in the codebook gives a limited CR of 15.06:1 while providing an acceptable execution time as well. Therefore, a large amount of redundancy still exists in the index stream generated from LBG VQ. In addition, Huffman encoding alone achieves very little improvement compared to LBG VQ of 1.6, which demonstrates that using an entropy coder alone is not sufficient if also using redundancy reduction before the entropy coder (e.g., LBG VQ).

Combining VQ with Huffman Encoding results in an approximately 22:1 CR. Due to the lossless nature of Huffman coding, the resulting image will remain the same as that produced by not encoding. Optimization and Delta Encoding have also been used to help improve compression efficiency through the correlation between the indices of two adjacent vectors. These optimizations have decreased entropy and produced a CR greater than 23, with a maximum of 23.45 utilizing filtering. The PSNR remains approximately 30 dB with an MSE value of around 64.75 and an average SSIM value of 0.8384, indicating that the original and reconstructed images maintain a very high structural similarity ratio. In addition, optimization of the VQ phase of the encode step has resulted in an approximate total execution time of 0.3368 seconds with no effect on performance. The results show that each of the components of the proposed hybrid framework contributes to its success, and all component methods combined achieve improved performance when compared to any component method performed by itself.

5.2.2 The effect of block size and codebook size

To examine the influence of block size on compression performance, experiments were conducted using different block sizes while fixing the codebook size at K = 64, as shown in Table 4.

Table 4. Effect of block size on compression performance at $K = 64$

Block size	PSNR (dB)	SSIM	System CR (:1)	LBG Time (s)
2 × 2	34.49	0.9209	7.03	0.4448
4 × 4	30.02	0.8384	23.45	0.2960
8 × 8	26.82	0.7494	35.80	0.1747

Note: CR = compression ratio, PSNR = peak signal-to-noise ratio, SSIM = structural similarity index measure, LBG = Linde–Buzo–Gray.

The 2 × 2 block size produced the best reconstruction quality, as indicated in Table 4, with an SSIM of 0.9209 and a PSNR of 34.49 dB. Nevertheless, a longer LBG execution time of 0.4448 s and a lower system CR of 7.03 accompanied this increase. In contrast, the 8 × 8 block size decreased the LBG execution time to 0.1747 s and achieved a higher system CR of 35.80×. However, the reconstruction quality decreased to an SSIM of 0.7494 and 26.82 dB in PSNR. With a PSNR of 30.02 dB, an SSIM of 0.8384, a system CR of 23.45×, and an LBG execution time of 0.2960 s, the 4 × 4 block size offered a balanced trade-off. To balance reconstruction quality, compression efficiency, and computational cost, a 4 × 4 block size was chosen for the proposed framework based on these findings.

Table 5 presents the PSNR values obtained for the tested images together with their corresponding reconstructed images. The reconstructed images provide a visual assessment of the compression results, while the PSNR values provide a quantitative measure of reconstruction quality

Table 5. Peak signal-to-noise ratio (PSNR) (dB) and reconstructed images

Image Name	PSNR	Reconstructed Image
Lena	30.07	
Cameraman	30.03	
Peppers	29.82	
Bout	28.87	

Table 6. Effect of codebook size on compression performance at a block size of 4 × 4

K	PSNR	SSIM	System CR	LBG Time
32	28.82	0.8054	30.65×	0.1205
64	30.02	0.8384	23.45×	0.2960
128	31.17	0.8650	18.18×	0.5862

Note: CR = compression ratio, PSNR = peak signal-to-noise ratio, LBG = Linde–Buzo–Gray.

Table 6 presents that reconstruction improved as the codebook size increased. PSNR and SSIM increased from 28.82 dB and 0.8054 at $K = 32$ to 31.17 dB and 0.8650 at $K = 128$, respectively. However, this improvement was accompanied by a decrease in the system CR from 30.65:1 to 18.18:1, and an increase in the LBG execution time from 0.1205 s to 0.5862 s. At $K = 64$, the framework achieved a PSNR of 30.02 dB, an SSIM of 0.8384, a system CR of 23.45×, and an LBG execution time of 0.2960 s at $K = 64$. As a result, K = 64 was selected as an intermediate configuration that provided a practical balance among reconstruction quality, compression efficiency, and computational cost.

5.3 Comparative results

Table 7 provides a summary of the new methodology

compared to many of today's image compression techniques using standard images such as Lena and Boat. The suggested approach was assessed using the same benchmark images and image resolutions supplied by the relevant research whenever feasible in order to increase the comparison's fairness. Nonetheless, variations in hardware settings and implementation specifics persisted among the approaches under comparison. According to the results obtained, the design of the proposed VQ image compression framework is not solely based on optimizing one performance metric. Rather, it is designed to provide a good balance between CR, image quality, and execution time; thus, making the VQ framework a practical and efficient option for those applications in which storage space and bandwidth are at a premium.

Table 7. The performance comparison analysis using standard Lena, Boat, and Cameraman images

References	Method	Resolution	CR	PSNR	Time	Image Name
[6]	DWT–Huffman Hybrid	512 × 512	1.52	54.28		Lena
[16]	2DWT + PCA + Canonical Huffman (CHC)	512 × 512	21.05	35		
Proposed	VQ-Delta coding-HC	512 × 512	23.45	30.02		
[2]	DWT–Huffman Hybrid	512 × 512	1.41	54.07		Boat
Proposed	VQ-Delta coding-HC	512 × 512	26.71	29.52		
[5]	2DWT + PCA + Canonical Huffman (CHC)	256 × 256	4.45	37.99		
Proposed	VQ-Delta coding-HC	256 × 256	12.80	24.54		
[17]	FLBG (Fast LBG with Pre-Scaling)	512 × 512	-	38-28	4.02 s	Cameraman
Proposed	VQ-Delta coding-HC	512 × 512	29.52	30.08	0.310	Cameraman

Note: CR = compression ratio, PSNR = peak signal-to-noise ratio, DWT = Discrete Wavelet Transform, VQ = vector quantization, LBG = Linde–Buzo–Gray, PCA = principal component analysis.

The DWT–Huffman method reported in the study [6] achieved a high-quality reconstruction of the Lena image (512 × 512) with a PSNR of 54.28 dB but a low CR, which indicates that this method places a much greater emphasis on obtaining a high-quality reconstruction than it does on achieving a high efficiency of compression. In contrast, the PCA–DWT–CHC algorithm developed in the study [16] produces a significantly higher CR while maintaining an acceptable level of reconstruction quality.

The proposed compression technique achieved the greatest CR of any of the compression methods tested for the Lena image, while maintaining acceptable quality of the reconstructed image. Although the PSNR is lower than that of the other transformation-based methods, the reconstructed image is visually acceptable. Additionally, the proposed framework obtained a lower execution time while keeping competitive reconstruction quality when compared to the recent FLBG technique [17]. However, variations in hardware platforms, software environments, and implementation parameters affect comparisons of execution times. An analogous result was also observed for the image of the boat. The method reported in the study [6] has the highest quality of reconstruction at 512 × 512 resolution, resulting in the lowest amount of CR possible. Whereas the proposed method achieves the highest amount of compression at a reduced computational complexity. To continue our comparison to the study [16], the boat image was judged according to the same experimental design as the study [16] at 256 × 256 resolution; however, under this configuration, the proposed method has a higher CR and still produces an acceptable reconstruction quality with an execution time of about 0.38 seconds (for the aforementioned experimental setup), indicating lower computational costs.

The enlarged view in Figure 4 confirms that the proposed framework preserves the main edge structures and visually important details after the compression process. Slight smoothing is observed in some fine-textured regions, which is consistent with the lossy nature of VQ. However, no noticeable blocking artifacts or severe visual distortions are observed, indicating that the reconstructed image maintains satisfactory visual quality. These visual observations are consistent with the quantitative PSNR and SSIM results.

Figure 4. Visual comparison of the original and reconstructed Lena images using an enlarged region of interest (ROI)

5.4 Effect of bilateral filtering on compression performance

The proposed compression framework was evaluated in two configurations: with and without bilateral filtering. To ensure a fair comparison, only the bilateral filtering stage was removed in the second configuration, while all compression parameters, benchmark images, and experimental conditions were kept unchanged. PSNR, SSIM, and MSE were used to assess the reconstruction quality. Table 8 presents the comparison results.

All four test images exhibit a steady improvement in reconstruction quality, according to the results. While the average SSIM climbed from 0.8266 to 0.8447, the average PSNR increased from 29.48 to 29.88 dB. Simultaneously, the average MSE dropped from 73.44 to 67.04. Lena, Cameraman, Peppers, and Boat all showed similar tendencies, suggesting that the progress wasn't limited to just one picture. These findings imply that despite retaining structural information important for reconstruction, adopting mild edge-preserving smoothing prior to VQ minimizes local intensity changes that

could impact codebook representation. Consequently, the bilateral filtering stage consistently improved reconstruction quality within the examined image set and compression parameters.

Table 8. Comparison of reconstruction quality with and without bilateral filtering

Image	PSNR (without)	PSNR (with)	SSIM (without)	SSIM (with)	MSE (without)	MSE (with)
Lena	29.67	30.02	0.8190	0.8384	70.13	64.75
Camera man	29.75	30.08	0.8855	0.8908	68.84	63.84
Peppers	29.39	29.88	0.7624	0.7967	74.90	66.89
Boat	29.11	29.52	0.8394	0.8528	79.88	72.66
Average	29.48	29.88	0.8266	0.8447	73.44	67.04

Note: PSNR = peak signal-to-noise ratio, MSE = mean squared error, SSIM = structural similarity index measure.

6. CONCLUSION

This paper presents a hybrid method for lossy compression of grayscale images based on traditional techniques. Combining VQ, Delta Coding, and Huffman coding, preceded by an edge-preserving bilateral filtering stage, exploits both spatial and statistical redundancy in the images.

Experimental evaluation using standard 512 × 512 grayscale benchmark images demonstrated that the proposed framework achieved an average CR of 26.09:1 while maintaining an average PSNR of 29.88 dB and an average SSIM of 0.8447. The average LBG execution time was 0.3104 seconds. The results demonstrate that Delta coding decreases the redundancy of VQ indices, while Huffman coding further improves compression efficiency without affecting reconstruction quality. Each stage contributes to improving the CR while preserving image quality.

Overall, the proposed framework provides a balanced trade-off between reconstruction quality, compression efficiency, and computational cost. These characteristics make the proposed framework suitable for image-intensive information systems that require efficient image storage and transmission under limited computational and storage resources. However, the present study was validated only on grayscale benchmark images; therefore, the reported results should be interpreted within the scope of the evaluated datasets. Future work will extend the evaluation to larger and more diverse datasets, including color, medical, and higher-resolution images, while also investigating parallel implementations to further improve computational efficiency.

REFERENCE

[1] Kulkarni, P., Dixit, M. (2025). Enhancing visual perception in image compression through an adaptive DCT-SVD hybrid algorithm. Results in Engineering, 28: 107205. https://doi.org/10.1016/j.rineng.2025.107205

[2] Vijaywargiya, Y., Pandey, R.K. (2025). Hybrid image compression algorithm based on singular value decomposition and discrete wavelet transform. Procedia Computer Science, 260: 1043-1051. https://doi.org/10.1016/j.procs.2025.03.289

[3] Abedi, F., Al-Baghdadi, A.F. (2024). Multi-objective optimization for vector quantization via genetic algorithm. Optics Continuum, 3(5): 808-822. https://doi.org/10.1364/optcon.517311

[4] Mao, Y., Wang, J., Guan, N., Xue, C.J. (2025). WISE: A framework for gigapixel whole-slide-image lossless compression. In 2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 29342-29351. https://doi.org/10.1109/cvpr52734.2025.02732

[5] Das, T., Choi, K. (2025). A study on advances in whole slide image compression. IEEE Access, 13: 202807-202823. https://doi.org/10.1109/access.2025.3637830

[6] Thomas, S., Krishna, A., Govind, S., Sahu, A.K. (2025). A novel image compression method using wavelet coefficients and Huffman coding. Journal of Engineering Research, 13(1): 361-370. https://doi.org/10.1016/j.jer.2023.08.015

[7] Zhu, Y., Huang, M., Zhu, Y., Zhang, Y. (2025). A low-complexity lossless compression method based on a code table for infrared images. Applied Sciences, 15(5): 2826. https://doi.org/10.3390/app15052826

[8] Ungureanu, V.I., Negirla, P., Korodi, A. (2024). Image-compression techniques: Classical and "region-of-interest-based" approaches presented in recent papers. Sensors, 24(3): 791. https://doi.org/10.3390/s24030791

[9] Abdulrazzaq, S.T., Siddeq, M.M., Zulkifley, M.A., Zaman, M.H.M., Moubark, A.M. (2026). Novel medical image compression/decompression technique based on bicubic interpolation with matrix reduction algorithm. IEEE Access, 14: 34600-34613. https://doi.org/10.1109/access.2026.3668851

[10] Chavan, P.P., Singh, M. (2024). Intelligent image compression model on the basis of wavelet transform and optimized fuzzy C-means-based vector quantisation. Journal of Information & Knowledge Management, 24(3): 2450050. https://doi.org/10.1142/s0219649224500503

[11] Xu, H., Wu, X., Zhang, X. (2025). Multirate neural image compression with adaptive lattice vector quantization. In 2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Nashville, USA, pp. 7633-7642. https://doi.org/10.1109/cvpr52734.2025.00715

[12] Zhang, X., Wu, X. (2023). LVQAC: Lattice vector quantization coupled with spatially adaptive companding for efficient learned image compression. In 2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Vancouver, Canada, pp. 10239-10248. https://doi.org/10.1109/cvpr52729.2023.00987

[13] Duan, Z., Lu, M., Ma, J., Huang, Y., Ma, Z., Zhu, F. (2024). QARV: Quantization-aware ResNet VAE for lossy image compression. IEEE Transactions on Pattern Analysis and Machine Intelligence, 46(1): 436-450. https://doi.org/10.1109/tpami.2023.3322904

[14] Puthentharayil Vikraman, B., Afthab, J. (2024). Effective image compression using hybrid DCT and hybrid capsule auto encoder for brain MR images. Journal of Visual Communication and Image Representation, 104: 104296. https://doi.org/10.1016/j.jvcir.2024.104296

[15] Pranitha, K., Kavya, G. (2023). An efficient image compression architecture based on optimized 9/7 wavelet transform with hybrid post processing and entropy encoder module. Microprocessors and Microsystems, 98: 104821. https://doi.org/10.1016/j.micpro.2023.104821

[16] Ranjan, R., Kumar, P. (2023). An improved image

compression algorithm using 2D DWT and PCA with canonical Huffman encoding. Entropy, 25(10): 1382. https://doi.org/10.3390/e25101382
[17] Bilal, M., Ullah, Z., Mujahid, O., Fouzder, T. (2024). Fast Linde–Buzo–Gray (FLBG) algorithm for image compression through rescaling using bilinear interpolation. Journal of Imaging, 10(5): 124. https://doi.org/10.3390/jimaging10050124
[18] Ghadami, R., Rahebi, J. (2023). Compression of images with a mathematical approach based on sine and cosine equations and vector quantization (VQ). Soft Computing, 27(22): 17291-17311. https://doi.org/10.1007/s00500-023-08060-9
[19] Camacho-Gonzalez, F.D., Lima-López, D., Zapotecas-Martínez, S., Altamirano-Robles, L. (2025). Vector quantization-driven image compression through multi-objective evolutionary algorithms. Expert Systems with Applications, 261: 125512. https://doi.org/10.1016/j.eswa.2024.125512
[20] Rahebi, J. (2022). Vector quantization using whale optimization algorithm for digital image compression. Multimedia Tools and Applications, 81(14): 20077-20103. https://doi.org/10.1007/s11042-022-11952-x
[21] Wu, M., Zhong, Q. (2024). Image enhancement algorithm combining histogram equalization and bilateral filtering. Systems and Soft Computing, 6: 200169. https://doi.org/10.1016/j.sasc.2024.200169
[22] Bo, L. (2025). Adaptive bilateral filter: A robust image denoising model based on local variance. In 2025 4th International Conference on Electronic Information Technology (EIT), Chengdu, China, pp. 589-593. https://doi.org/10.1109/eit67313.2025.11231922
[23] Yang, Y., Sun, Y., Gao, W., Wang, X., Zeng, L. (2024). Bilateral regularized optimization model for edge-preserving image smoothing. Image and Vision Computing, 146: 105031. https://doi.org/10.1016/j.imavis.2024.105031
[24] Lin, Y., Liu, J.C., Chang, C.C., Chang, C.C. (2024). Lossless recompression of vector quantization index table for texture images based on adaptive Huffman coding through multi-type processing. Symmetry, 16(11): 1419. https://doi.org/10.3390/sym16111419
[25] Zhu, Y., Liu, Y., Zhu, Y., Huang, M., Jiang, J., Zhang, Y. (2025). Lossy infrared image compression based on wavelet coefficient probability modeling and run-length-enhanced Huffman coding. Sensors, 25(8): 2491. https://doi.org/10.3390/s25082491
[26] Han, S., Mo, B., Zhao, J., Xu, J., Sun, S., Jin, B. (2024). High-quality image compression algorithm design based on unsupervised learning. Sensors, 24(20): 6503. https://doi.org/10.3390/s24206503
[27] Mohammadi, S., Jenadeleh, M., Sneyers, J., Saupe, D., Ascenso, J. (2026). Evaluation of objective image quality metrics for high-fidelity image compression. IEEE Access, 14: 35651-35668. https://doi.org/10.1109/access.2026.3669417

NOMENCLATURE

CR	Compression Ratio
PSNR	Peak Signal-to-Noise Ratio
MSE	Mean Squared Error
SSIM	Structure Similarity Index Measure
σ_r	Range Parameter
σ_s	Spatial Parameter
Δ	Delta Defference
μ_x	Intensty of x
μ_y	Intensty of y
σ_x^2	Variance of x
σ_y^2	Variance of y