CHAPTER 3
WAVELET BASED ENHANCED COLOR IMAGE COMPRESSION USING SUBBAND VECTOR QUANTIZATION

3.1. INTRODUCTION

Increase in the use of color images in the continuous expansion of multimedia applications has increased the demand for efficient techniques that can store and transmit visual information. This demand has made image compression a vital factor and has increased the need for efficient algorithms that can result in high compression ratio with minimum loss. This chapter proposes an innovative technique for compressing color still images using wavelet compression scheme. The wavelet transform has been successfully used in image coding since it allows localization in both the space and frequency domains [15]. The proposed system uses DWT to improve compression technique in terms of compression ratio and image quality. The system is developed in such a way that it should work for color still images. This is achieved by concentrating on the use of predictors for subbands across different color components based on binary vector morphology.

The main objective of the present research work is to propose and develop a system that can perform compression on color image efficiently. To achieve the above primary objective, the following aims were formulated.

- To develop an efficient and effective color image compression technique using discrete wavelet transformation.
- The proposed system uses the following techniques:
  - Subband coding to optimize the process of image compression.
  - Wavelet tree structured vector quantization for lower subbands. This technique uses VQ and a modified LBG (Linde, Buzo and Gray) algorithm to cluster the codebooks.
  - Region growing prediction algorithm to lower subbands
• Use of morphological predictors for higher subbands across different color components based on binary vector morphology.
• Use of Bit Plane Coding (BPC) and Binary Arithmetic Coding (BAC) to encode the coefficients derived.

- To compare the proposed system with a standard technique.

To achieve the above objectives, the proposed scheme uses wavelet transformation, tree structured vector quantization and binary vector morphological prediction for compressing color images. Binary vector morphology is used to predict the significance of coefficients in the subbands across different color components. The use of tree structured vector quantization reduces the search time for quantization and coding. This greatly enhanced the proposed algorithm in terms of compression and decompression time.

The chapter is organized as follows. Section 3.1 explains the main techniques used. Section 3.2 explains the methodology followed by the proposed algorithm and the experimental results obtained are explained in Section 3.3. Section 3.4 summarizes the work and presents future research directions.

3.2. TECHNIQUES USED

This section gives a brief overview of the preliminary concepts used in the proposed work.

3.2.1. Color Space Transformation

A color model is an abstract mathematical model describing the way colors can be represented as tuples of numbers, typically as three or four values or color components (e.g. RGB and CMYK are color models) and the mapping of colors from the color model is the color space. The two most widely used color spaces for storing digital images are RGB color space and YUV color space.
The RGB color space is defined by the three chromaticities of the red, green, and blue additive primaries, and can produce any chromaticity. RGB stores a color value for each color levels, Red, Green and Blue. It uses 8 bits to each color level, thus using 24 bits to store a pixel color information.

YUV color space works on the principle that as human eye perceives changes in brightness better than changes in color, focus more on brightness than the actual brightness level. There are three values while using YUV color space [46]. They are,

- Luminance (Luma, which is the brightness level) is abbreviated as $Y$.
- $U$ is the red difference sample
- $V$ is the blue difference sample

YUV color space is stored using 16 bits, where 8 bits are used by Luma and 4 bits are used by each of $U$ and $V$. Thus, YUV stores more relevant data at a lower accuracy than RGB. Moreover, it is well known that the RGB components of color images are highly correlated and if the wavelet transforms of each color component is obtained, the transformed components will also be highly correlated [77]. Therefore, a color transformation that reduces the psychovisual redundancy and correlation of the image is highly desired. Many linear transformations can be used such as the YUV, KLT, YCrCb or La*b*. The present research work uses a YUV color transformation for the following reasons:

1. The YUV color model represents the human perception of color more closely than the standard RGB model used in computer graphics hardware and

2. The YUV color model stores more relevant data at a lower accuracy than RGB.

The general formula for the color space transformation is given in Equation 3.1.
\[
\begin{bmatrix}
S_1' \\
S_2' \\
S_3'
\end{bmatrix} =
\begin{bmatrix}
C_{11} & C_{12} & C_{13} \\
C_{21} & C_{22} & C_{23} \\
C_{31} & C_{32} & C_{33}
\end{bmatrix}
\begin{bmatrix}
S_1 \\
S_2 \\
S_3
\end{bmatrix}
\]  

(3.1)

where \([S_1, S_2, S_3]^T\) is the original color space, \([S_1', S_2', S_3']^T\) is the transformed color space and \([C_{ij}]\) is the coefficient matrix of transformation.

Figure 3.1 shows an example image in the YUV color space and each of the three components. This illustrates the advantage of using the YUV color space, where most of the information is contained in the luminance space (Figure 3.1b).

\[\begin{align*}
Y &= 0.200 \ Y + 0.587 \ U + 0.114 \ V \\
U &= -0.418 \ Y - 0.289 \ U + 0.437 \ V \\
V &= 0.615 \ Y - 0.515 \ U - 0.100 \ V
\end{align*}\]  

(3.2)
3.2.2. Wavelet subband decomposition

Wavelet is a mathematical function that is used to divide a given signal into different frequency components which can be studied individually with a resolution that matches its scale. A wavelet transform is the representation of a function by wavelets. Wavelet transforms have advantages over traditional cosine and Fourier transforms for representing functions that have discontinuities and sharp peaks, and for accurately deconstructing and reconstructing finite, non-periodic and/or non-stationary signals [115].

Discrete Wavelet analysis is computed using the concept of filter banks. Filters of different cut-off frequencies analyze the signal at different scales. Resolution is changed by the filtering and the scale is changed by up sampling and down sampling. If a signal is put through two filters,

i. a high pass filter - high frequency information is kept, low frequency information is lost.

ii. a low pass filter - low frequency information is kept, high frequency information is lost.

then the signal is effectively decomposed into two parts, a detailed part (high frequency), and an approximation part (low frequency). The subsignal produced from the low filter will have a highest frequency equal to half that of the original [105]. According to Nyquist sampling, this change in frequency range means that only half of the original samples need to be kept in order to perfectly reconstruct the signal. More specifically this means that up sampling can be used to remove every second sample. The approximation subsignal can then be put through a filter bank, and this is repeated until the required level of decomposition has been reached. The ideas are shown in Figure 3.2.
The DWT is obtained by collecting together the coefficients of the final approximation subsignal and all the detail subsignals and is given by Equation 3.3.

\[ W_t(a, b) = \int_{-\infty}^{\infty} x(t) \Psi_{a,b}(t) dt \] (3.3)

Overall the filters have the effect of separating out finer and finer detail and if all these details are 'added' together then the original signal is reproduced.

The wavelet transform produces as many coefficients as there are pixels in the image. These coefficients can then be compressed more easily because the information is statistically concentrated in just a few coefficients. This principle is called transform coding. After that, the coefficients are quantized and the quantized values are entropy encoded.
Subband decomposition has the main advantage of dividing an image into layers with different frequencies [5]. Different coding schemes can then be applied to each subband to achieve maximum efficiency. Another property of subband decomposition is that the original image can be reconstructed from the subbands with minimum distortion. They also allow localization in both space and frequency domains with which the characteristics of the wavelet coefficients can be analyzed to achieve better efficiency. To take advantage of these properties, this research uses subband decomposition and exploits inter and intra subband correlations among the coefficients.

3.2.3. Quantization

Quantization is a many-to-one mapping that replaces a set of values with only one representative value. By definition, this scheme is lossy, because after mapping, the original value cannot be recovered exactly. There are two basic types of quantization,

(a) Scalar quantization and
(b) Vector quantization.

Scalar quantization (SQ) performs many-to-one mapping on each value, for example, it may store only the 6 most significant bits from 8 bit values. Vector Quantization (VQ) is the best way of quantizing and compressing images. It replaces arrays of values (i.e., blocks of pixels) with one value, which is the index from "codebook". The same index can be used to represent slightly different arrays of values; therefore it results in a lossy many-to-one mapping. The main implementation issues in the design of VQ algorithm is the

- Codebook design and storage of the codebooks in the image and
- Search Optimization - Computation complexity and time during the search for optimum code vector

When the codebook size is large, the reconstructed image will be very similar to the original image. At the other end, the image obtained using a small
sized codebook will contain a lot of visible artifacts. The size of the codebook is also important while calculating the transmission overhead, which has to be kept at a minimum. The storage of codebooks in the image is another important factor, which increases with the size of file. Smaller codebook size produces small transmission and storage overheads but degrades the quality of the reconstructed image.

- **Codebook Design**

A widely used algorithm for codebook design is the LBG (Linde-Buzo-Gray) algorithm, which is a vector quantization algorithm to derive a good codebook. It is similar to the k-means method in data clustering [61]. The algorithm is known as the Generalized Lloyd Algorithm (GLA) or simply LBG (Linde, Buzo and Gray) from the initials of its authors.

The generalized Lloyd algorithm is a clustering technique, extension of the scalar case [62]. It consists of a number of iterations, each one recomputing the set of more appropriate partitions of the input states (vectors), and their centroids. The algorithm is shown in Figure 3.3. It takes as input a set $T$ of $M$ input states, and generates as output the set $C$ of $N$ new states (quantization levels).

There are three design decisions to be made when using such technique:

**Stopping criterion**

Usually, average distortion of codebook at cycle $m$, $D_m$, is computed and compared to a threshold $0 (0 < 0 < 1)$ as in the following equation.

$$\frac{(D_m - D_{m+1})}{D_m} < 0 \quad (3.4)$$
1. Begin with an initial codebook $C_1$.

2. Repeat

   (a) Given a codebook (set of clusters defined by their centroids) $C_m = \{y_i, i = 1, \ldots, N\}$, redistribute each vector (state) $x \in T$ into one of the clusters in $C_m$ by selecting the one whose centroid is closer to $x$.

   (b) Recompute the centroids for each cluster just created, using the centroid equation given below to obtain the new codebook $C_{m+1}$.

   $$\text{cent}(R)[i] = \frac{1}{\|R\|} \sum_{j=1}^{\|R\|} x_{j}[i]$$

   where $x_j \in R$, $x_{j}[i]$ is the value of attribute $i$ of vector $x_j$, and $\|R\|$ is the cardinality of $R$.

   (c) If an empty cluster was generated in the previous step, an alternative code vector assignment is made (instead of the centroid computation).

   (d) Compute the average distortion for $C_{m+1}$ as $D_{m+1}$ until the distortion has only changed by a small enough amount since last iteration.

Figure 3.3: The Generalized Lloyd Algorithm

Empty cells

One of the most used mechanisms consists of splitting other partitions, and reassigning the new partition to the empty one. In each iteration, all empty cells generated by the GLA are changed by another cell. To define the new one, another non-empty cell with big average distortion $y$, is split into:

$$y_1 = \{y[1] - \varepsilon, \ldots, y[K] - \varepsilon\} \text{ and } y_2 = \{y[1] + \varepsilon, \ldots, y[K] + \varepsilon\}$$

Initial codebook generation

The main drawback of GLA algorithm is the initial codebook assumption. A bad initialization could lead to an impossibility of finding a good quantizer. For example, let us examine Figure 3.4. In Figure 3.4.a, the codeword number 4 will always generate an empty cell because all the elements of the data set are nearer to the other codewords. So, following the steps of the traditional LBG, it cannot move and will never represent any element, thus becomes useless. The same authors of the LBG [61] proposed solutions to this problem such as the assigning of the codeword to a non-empty cell.
To solve the problem of empty cell and initial codebook selection, a modified version of GLA that requires a partition split mechanism is described in the algorithm in Figure 3.5.

1. Begin with an initial codebook $C_1$ with $N$ (number of levels of the codebook) set to 1. The only level of the codebook is the centroid of the input.

2. Repeat
   (a) Set $N$ to $N \times 2$
   (b) Generate a new codebook $C_{m+1}$ with $N$ levels that includes the codebook $C_m$. The rest $N$ undefined levels can be initialized to 0
   (c) Execute the GLA algorithm with the splitting mechanism with parameters $(T, N)$ over the codebook obtained in previous step
      Until $N$ is the desired level

Figure 3.5 : A Version of the Generalized Lloyd Algorithm that Solves the Initial Codebook and Empty Cell Problems

The above algorithm also has some drawbacks. Figure 3.4b shows two clusters and three codewords. In the little cluster there are two codewords whereas, in the other, only one. The elements in the data set in the smaller cluster are all well approximated by the two related codewords. Instead, a lot of elements in the larger one are badly approximated by the related codeword. For this geometrical distribution, it would be preferable that two codewords were
inside the big cluster and only one in the other, but the LBG optimization
algorithm, in this situation, does not permit the migration of a codeword from
the little cluster to the big one. This is a great limitation. To solve this
drawback, the present research work introduces a modified GLA (MGLA)
proposed by [78]. The General working of the algorithm is shown in Figure
3.6.

In the MLGB algorithm, a new step called ELGB block was inserted
between Voronoi partition calculation and the calculation of the codebook
verifying the CC. The main function was to identify the possible situations of
local minima and to remedy them. On particular, some sub-optimal solutions
were adopted to keep the overhead introduced by the ELGB block low. More
detailed explanation with implementation details is given in [78, 79].

- Search Optimization - TSVQ

After codebooks are generated, the next step is to optimize the codebook
search process. There is no standard algorithm for codebook searching, so
different schemes are currently in use (for example, Full Search VQ, Tree-
Structured VQ, Pruned Tree-Structured VQ, Entropy-Pruned Tree-Structured
VQ, Entropy-Constrained VQ, etc.). In this research work TSVQ is used and is
described below.

TSVQ reduces the quantizer search complexity by replacing full search
encoding with a sequence of tree decisions. The idea behind TSVQ is that for a
VQ with \( L = K^d \) output points, it may be possible to break the search for the
best output vector for a given input down to a series of K searches in a tree
structure (Figure 3.7). Therefore, at each stage in the tree, a set of K vectors,
each of which can be considered representative of a set of vectors at the next
level in the tree, are needed. The encoded search is simplified because at any
level in the tree only K comparisons are performed, and the total number of
comparisons required to determine the best output vector is \( K^d \). When
contrasted with the $L = K^d$ comparisons needed for a full search, the reduction in complexity is evident.

![Flowchart of the Modified LGB Algorithm](image-url)

Figure 3.6: Modified LGB Algorithm
Normally, a TSVQ tree is grown by successively splitting nodes and then optimally pruning them until the desired rate is reached. In the present research work, a greedy method described in [9] is used to construct the tree along with the MLGB algorithm explained above. The basic problem here is whether the splitting should be done in the current layer or down to a new layer.

![Tree Structured Vector Quantization](Image)

**Figure 3.7 : Tree Structured Vector Quantization**

The representative code vectors is first partitioned into non-overlapping clusters by repeatedly applying the MLBG algorithm. A new vector is found near the vector for the cell to be split and is added to the vectors previously used for MLBG. MLBG is applied to the entire population of data vectors, again using the coarsest representation of each vector. These steps are repeated until the percentage reduction in distortion for the entire population falls below a predetermined threshold. Then the partition in the coarsest resolution is fixed, and further partitioning continues by splitting the existing cells based on finer
For a given block I which contains J cells at scale M, the average distortion is computed using the formula in Equation 3.6.

\[
D_{I,J}^{M} = \frac{\sum_{j=1}^{J} \sum_{x_{1j}^{M} \in \text{cell}_{j}} \left\| x_{1j}^{M} - m_{1,j}^{M} \right\|^2}{N}
\]

(3.6)

where \( m_{1,j}^{M} \) is the centroid of cell \( j \) at scale \( M \) and \( N \) is the total number of observations.

Compute \( \Delta D_{I,J}^{M} = \frac{D_{I,J-1}^{M} - D_{I,J}^{M}}{D_{I,J-1}^{M}} \). If it is larger than a prefixed threshold, than new centroid \( J+1 \) is added at the same scale, otherwise goes down to scale \( m+1 \).

After the training stage, all cells are labeled using majority voting, i.e., if class \( k \) dominate cell \( j \), then all samples falling into cell \( j \) will be classified as class \( k \). After the above training procedure, a hierarchical multi-resolution classifier is available.

In order to decrease the computation time, after the standard design procedure, the representative vectors (each representing \( n \) vectors) are replaced by their lower wavelet transform bands. An \( m \) stage TSVQ structure possesses \( m^n \) codevectors. For the \( k^{th} \) stage, each representative vector is replaced by the lower bands of the \( (m-k)^{th} \) wavelet transforms which have dimensions of \( ab/2^{2(m-k-1)} \) for the codevectors with original size of \( ab \). Further simplification is realized during the design stage. Cluster creation starts with the original codevectors and is combined into lowest group clusters. After obtaining the centroids of the clusters, the last stage representatives are obtained. Then the first stage wavelet transforms of the representative vectors are taken off and the lower bands are used as new representatives. Proceeding similarly gives the whole structure with less computation.
3.2.4. Prediction of significance of coefficients in the subbands

After optimization process, the next step is the actual search of the correct cluster that has the significant coefficient. For this purpose, a significance map is used, which is a binary representation of the “significance” of the wavelet coefficients relative to a given threshold. All coefficients that are greater than the threshold are said to be significant. The significance map can be thought of as a mask that indicates to the decoder the location of these "significant coefficients." While Shapiro encodes a series of these maps using zerotrees, in [64], a method for encoding a gray scale image is proposed by predicting the significance map of higher subband coefficients based on the significance of coefficients from lower subbands. This approach exploits the inter-band dependencies of the significant coefficients. The prediction of higher subband significance maps is done using binary morphological filters. The significance map for the lower subbands (coarser scale) needs to be encoded so that the higher subband significance maps can then be predicted. The lower subbands are encoded by using a region growing method, exploiting the intraband dependencies of the subband, the prediction of the other subbands is obtained by using vector morphological filters. The region growing method and vector morphological filters are explained in the following sections.

• Region Growing Spatial Prediction

The Region growing algorithm is used as a prediction algorithm for image compression in the present work. The reason for having such an encoding technique is to achieve better coding performances by adaptively exploiting spatial redundancies.

The spatial prediction can be summarized as follows: in order to predict a pixel value, consider its neighbors in a causal manner (left and top side depending on the scan sequence), combine them in order to produce an estimate of the present pixel value, round this prediction to the nearest integer and subtract it from the actual value. The obtained error is called prediction.
error. Prediction algorithms differ in the way the neighbors are combined. For instance, JPEG may combine the neighbors in 7 different ways [35]. Performances of each predictor depend upon image structure.

In the proposed research work, spatial prediction is performed from pixel neighbors belonging to the same region in order to avoid an erroneous estimation at region boundaries. The problem is that, due to the arbitrary shape of the regions, classical scanning sequences (as top-right/bottom-left) do not assure a correct prediction for all pixels in the region. To solve this, a region growing algorithm is proposed. Pixels inside the region are “scanned” in the same way as in region growing but only the neighbor pixels that have already been scanned are used for the spatial prediction. The process is illustrated in Figure 3.8 and is summarized in the following steps:

i) define the spatial regions \( r_1, r_2, \ldots, r_n \)

ii) locate the first point belonging to \( r_1 \), put its value on the final bit-stream as no prediction is possible for the first point.

iii) put its 4-connected neighbors that belong to \( r_1 \) on a queue and label them as “stacked”. Label the current pixel as “scanned” (dark region);

iv) extract the next pixel from the queue in a FIFO manner, look at the 4-connected neighbors and make a prediction from the ones that are labeled as “scanned”. Round the prediction to the nearest integer and compute the prediction error;

v) Go to (iii) until all pixels of \( r_1 \) are scanned;

vi) Repeat the procedure for the other regions.

In the illustrated example of Figure 3.8, pixel 12 is about to be predicted, its 4-connected neighbors are stored in the stack (pixels 19 and 18, 17 was already stacked by pixel 11 and 7 is labeled as “scanned”). The prediction is produced from the scanned neighbor (pixel 7).
With the presented method, spatial prediction is always performed from neighboring pixels belonging to the same region. The number of neighbors used for prediction is thus variable depending on the pixel position inside a region. Neighbor connectivity can be extended to 8 and the same method applies.

- **Vector Morphology**

  Mathematical morphology is one of the important techniques that have been used in different image processing applications. It is a geometric approach that has been developed as a powerful tool for shape analysis in binary and grayscale images. Unfortunately, the extension of the concepts of binary and grayscale morphology to color images is not a straightforward task [64].

  In morphological sense, the term filter is restricted to all image-image transformations that are translation-invariant and increasing. The definition of morphological filter or M-filter sometimes requires idempotency in addition to the above requirement of filter.

  Morphological filters are nonlinear signal operators that locally modify the geometrical features of a signal. Given a binary image X, and a 2-D binary structuring element B, the dilated image is defined as the union of all the pixels that fall under B when it is centered at each pixel in X. The eroded image is similarly defined as the intersection of pixels that fall under B when centered at
each pixel in X. These two operations are obtained using equation (3.7) and (3.8) respectively.

\[
D(A, B) (r, s) = \max_{(j,k) \in B} (A(r-j, s-k) + B(j, k))
\]  

(3.7)

\[
E(A, B) (r, s) = \min_{(j,k) \in B} (A(r+j, s+k) - B(j, k))
\]  

(3.8)

where A is the image and B is the structural element. These two operations are illustrated in the following Figure 3.9.

![Figure 3.9: A binary image containing two object sets A and B. The three pixels in B are "color-coded" as is their effect in the result](image)

(a) Dilation D(A, B)  
(b) Erosion E(A, B)

Figure 3.9 : A binary image containing two object sets A and B. The three pixels in B are "color-coded" as is their effect in the result

Dilation, in general, causes objects to dilate or grow in size; erosion causes objects to shrink. The amount and the way that they grow or shrink depend upon the choice of the structuring element.

Given a vector valued binary image Y, morphological filtering can be defined by using component-wise operators. In this case, the structuring element can then be different for each component. The concept of morphological vector filters can be extended by using a scalar valued function that maps each vector \( y_j \) to a scalar value \( d_j \). In the present work, for a three component binary valued vector the mapping is \( d: \{0,1\}^3 \to \mathbb{R} \) [26]. Dilation is then the binary valued vector under the structuring element that has maximum \( d \), and erosion would similarly, be the vector with the minimum \( d \). Different choices of \( d \) can potentially lead to different results. The component-wise approach for vector valued images is generally referred in the literature as
marginal ordering, and the approach using the mapping is known as reduced
ordering [119].

The function of morphological filters is to enlarge or reduce the area
where significant coefficients are located in the image. The prediction indicates
the location of the significant coefficient. All coefficients not covered by the
prediction need to be encoded separately. The concept of applying
morphological predictors for inter-band dependencies was proposed by [86]
and is applied in the proposed algorithm.

- **Quality factor**

  Another factor that affects the compression performance is the usage of
quality factor. The quality factor is a number that determines the degree of loss
in the compression process. It is always desirable to have an image
compression algorithm that is suitable for a wide range of applications. To
achieve this, one useful property that can be included is to allow adjusting
compression parameters (quality factor) to increase or decrease the degree of
lossiness. This provides a trade-off between compression rate and image
quality: the higher the setting, the better the quality of the resultant image but at
the cost of a larger file size. Quality factors (QF) are used during TSVQ
process. Setting the quality factor will have direct effect on the amount of
quantization performed with a trade off against compression ratio and image
quality. The QF between 0-100 is the widely accepted scale range.

3.3. METHODOLOGY

In the following sections, the first proposed approach “wavelet based
enhanced color image compression approach using subband vector
quantization” is explained. The main objective of this approach is to reduce
both the space requirements as well as time for processing. The block diagram
of the proposed approach is shown in Figure 3.10. The three basic steps
involved in this approach are:
Step 1: Color Space Transformation

Step 2: Wavelet subband decomposition
   a) Vector Quantization, clustering and search optimization algorithm
   b) Region Growing prediction
   c) Morphological Predictors

Step 3: Entropy coder

The methodology of the proposed system is described as below:

As the first step, a color transformation is performed in order to decorrelate the color components, followed by the actual compression scheme.

Figure 3.10: Image Compression System
The first stage, subband decomposition is done through the two-dimensional wavelet transform on the image. The Daubechies family of basis functions for DWT, which is widely used in image compression [21], is used for this purpose. After the decomposition process, the energy subbands obtained are given below and an example is shown in Figure 3.11.

- Low-Low (LL)
- Low-High (LH)
- High-Low (HL) and
- High-High (HH).

![Figure 3.11: Level 1 Wavelet Transform of Lena](image)

As it can be seen from the figure, the least energy containing the most redundant band is HH band and it is treated as Gaussian noise [114] and therefore is ignored with minimum loss of information.

The LH (lower subband) and HL (higher subband) bands also exhibit the characteristics of a high frequency signal; but there exists correlation among the horizontal and vertical pixels for the former and latter bands, respectively. Using classical transform coding techniques such as the discrete cosine transform (DCT) [54], for these bands are useless attempt due to the fact that the coefficient distribution of the resultant transformed matrix does not exhibit
localization of high energy coefficients as expected in general. The main reason behind this phenomenon is the absence of high correlation between the pixels.

Due to the reasons stated above, quantization should be performed before compressing these bands, unless they are dismissed as noise. Usually, scalar quantization followed by entropy coding is applied to these bands as the quantization-compression scheme. However, for a variable sized codeword, VQ should give promising results when the codebook is trained effectively. As correlation exists in horizontal or vertical directions for these bands, by choosing appropriate codevector sizes, vector quantization can be used most efficiently. A construction of the codevectors as suggested by [5] is utilized in order to exploit the interpixel dependencies in these bands. The authors suggest a 4x8 blocks for the LH band, and 8x4 blocks for the HL parts in the vector quantizer code vectors. The same codebook is employed for both bands for memory considerations, but each code vector is also used for another band by taking the transpose of it in order to obtain the appropriate size. This saves from needing two different codebooks, which would double the memory allocation requirement of the scheme. Figure 3.12 shows this in Lena image subbands. In the LH and HL bands of the image, similar patterns are observed which can be coded by the same code vectors.

![Figure 3.12: Similarities between subbands](image)

After discarding the HH band and coding the HL and LH subbands, the LL band is further decomposed (Figure 3.13). This does not bring together much
computational load, because the size of the LL band is one fourth of the original image. At this point, the LL band is replaced by the following bands

(a) Low-Low-Low (LLL)
(b) Low-High-Low (LHL)
(c) Low-Low-High (LLH) and
(d) Low-High-High (LHH).

LHH band contains significant amount of energy and cannot be discarded.

![Figure 3.13 : Second Level Decomposition of Lena](image)

The same quantization scheme described earlier was applied to all four high frequency bands of the second level transform, namely LLH, LHL and LHH. 4x8 blocks used for the HL and LH bands are wavelet transformed to 2 x 4 blocks for usage in LHL, LLH and LHH. Although it may look as if a separate codebook consisting of 2x4 codevectors was generated, the difference lies in the training and coding stages. The 4x8 and 2x4 blocks are in effect simultaneously trained since the codevectors are related through the wavelet transform. The only additional computation is in the training stage, where the DWT of the codevectors are calculated. After deciding on the codebook size,
the optimized wavelet based TSVQ described earlier was used as search algorithm. Here, during the construction of the tree structure, the vectors are combined according to the Euclidean distance between bands of their wavelet transforms. Thus, the LL band of the codevector becomes the representative for that vector. In fact, WTSVQ can be interpreted as a modified version of the classified VQ scheme. This approach reduces the computational complexity as the number of vectors is \( m/2 \times n/2 \) vectors rather than \( m \times n \). One other advantage of WTSVQ is its capability of decreasing the codevector dimensions. As a result, when clustering the vectors, the reduction in the dimension of the vector space enables very effective clustering, which lead to more efficient trees.

The tree is constructed in two stages. In the first stage, there are ‘\( p \)’ representative codewords. Below this layer, there are ‘\( r \)’ additional codevectors related to each of the ‘\( p \)’ first stage codevectors, giving rise to a codebook of size ‘\( rp \)’. After the tree structure is set up, the elements of the training set is used to develop the codebook. The first subblock of the training set matrix is taken and is compared with the ‘\( p \)’ first stage elements, resulting in an index of the best-fitting element. Then, the training set subvector is compared with the ‘\( r \)’ elements which lie below the first stage’s best fitting vector. This process requires \( r + p \) comparisons for determining the best fitting codeword instead of \( rp \).

Finally, as the LLL subband has highly correlated pixels and exhibit Gaussian distribution in their histograms, a deadband scalar quantization is performed.

Last part of the compression scheme involves the coding of the LLL band. Although there is high correlation among the pixels of this band, the DCT coefficients are not localized enough. So a deadband scalar quantization is used to meet all coefficients to a centroid threshold zero. All coefficients close to zero are discarded and a good compression ratio is achieved.
The final stage of the proposed system is encoding. The 2nd level coefficients of the LL subbands are entropy encoded separately. The rest of the coefficients are encoded from coarse to fine scales with each color component image treated separately. The lower subbands are then encoded independently using a region-growing algorithm described earlier, which consisted of scanning a neighborhood of significant pixels and growing this neighborhood iteratively as long as it includes significant pixels. In order to start the prediction of the higher order subbands, a structuring element and a vector morphological filter need to be chosen as described previously. The morphological filters enlarges or reduces the area where the scanning should take place (the prediction indicates the location of the significant coefficients). All coefficients not covered by the prediction maps are encoded separately.

The entropy encoder used are Bit Plane Coding (BPC) and Binary Arithmetic Coding (BAC) [106]. The combination of BPC and BAC is referred to as Tier 1 coding. BPC has three passes in each bit plane: Significance Propagation Pass, Magnitude Refinement Pass, and Cleanup Pass. Each pass generates context models and the corresponding binary data. The output of BPC and BAC produces the compressed bit stream. So each coding block has an independent bit stream. These independent bit streams of all the code blocks are combined into a single bit stream using Tier 2 coding, which is based on the result of rate-distortion optimization. Tier 2 coding multiplexes these independent bit streams that were generated in Tier 1 coding to compose the final compressed output bit stream. It also efficiently gives header information to indicate ordering of the resulting coded blocks and corresponding coding passes.

3.4. EXPERIMENTAL RESULTS

The proposed system was vigorously tested with test images and was evaluated using the results obtained.
3.4.1. Test Images

The original color images are all of size 256x256 and the system was tested with varying quality factors ranging from 10-100 in steps of 10. The results projected in this chapter, were obtained by examining the proposed system with the images shown in Figure 3.14.

3.4.2. Results

The proposed system was evaluated in terms of quality metrics like compression ratio, compression and decompression time and Peak Signal to Noise Ratio (PSNR). To authenticate the proposed system, the results are compared with the standard JPEG 2000 image compression technique. The results are discussed under the following headings.

- Compression ratio
- Speed of Compression
- Quality of decompression
- PSNR and Compression Ratio

a) Compression Ratio

The compression performances of the proposed algorithm for the selected images are shown in Figure 3.15 and are tabulated in Table 3.1.

<table>
<thead>
<tr>
<th>Image</th>
<th>QF 10</th>
<th>QF 20</th>
<th>QF 30</th>
<th>QF 40</th>
<th>QF 50</th>
<th>QF 60</th>
<th>QF 70</th>
<th>QF 80</th>
<th>QF 90</th>
<th>QF 100</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lena</td>
<td>53.08</td>
<td>51.83</td>
<td>50.14</td>
<td>48.02</td>
<td>45.32</td>
<td>41.57</td>
<td>36.98</td>
<td>31.20</td>
<td>23.87</td>
<td>14.21</td>
</tr>
<tr>
<td>Pepper</td>
<td>48.38</td>
<td>46.90</td>
<td>45.11</td>
<td>43.08</td>
<td>40.64</td>
<td>37.50</td>
<td>33.51</td>
<td>28.56</td>
<td>22.42</td>
<td>14.38</td>
</tr>
<tr>
<td>Zelda</td>
<td>55.95</td>
<td>54.81</td>
<td>53.54</td>
<td>51.73</td>
<td>49.45</td>
<td>46.7</td>
<td>42.86</td>
<td>37.29</td>
<td>30.01</td>
<td>19.64</td>
</tr>
<tr>
<td>Barbara</td>
<td>56.80</td>
<td>53.76</td>
<td>50.86</td>
<td>47.60</td>
<td>43.62</td>
<td>38.89</td>
<td>34.00</td>
<td>28.43</td>
<td>21.73</td>
<td>13.67</td>
</tr>
</tbody>
</table>
Figure 3.14: Test Images Used

(a) Lena

(b) Pepper

(c) Zelda

(d) Barbara
As seen from the data, the analysis revealed that with decreasing quality factor, the compression ratio increases. Our results are in agreement with the reports of [58] and [40] reiterating the efficiency in performance of the proposed algorithm.

Figure 3.16 and Table 3.2 show the comparison of the proposed system with JPEG 2000 for the image Zelda (Figure 3.14c) taking Quality factor Vs compression ratio into consideration.

**TABLE 3.2**

<table>
<thead>
<tr>
<th>QF</th>
<th>10</th>
<th>20</th>
<th>30</th>
<th>40</th>
<th>50</th>
<th>60</th>
<th>70</th>
<th>80</th>
<th>90</th>
<th>100</th>
</tr>
</thead>
<tbody>
<tr>
<td>Proposed</td>
<td>55.95</td>
<td>54.81</td>
<td>53.52</td>
<td>51.75</td>
<td>49.48</td>
<td>46.70</td>
<td>42.86</td>
<td>37.29</td>
<td>30.07</td>
<td>19.64</td>
</tr>
<tr>
<td>JPEG 2000</td>
<td>53.99</td>
<td>53.26</td>
<td>52.70</td>
<td>49.26</td>
<td>48.00</td>
<td>44.90</td>
<td>40.20</td>
<td>35.43</td>
<td>28.76</td>
<td>18.00</td>
</tr>
</tbody>
</table>

As it is evident from the Table data and Figure, the proposed system just outperforms the JPEG 2000 technique with an average increase of 3.98% in the compression ratio of the proposed system when compared with JPEG 2000.

**b) Compression and Decompression Time**

The main goal of any compression algorithm is to maximize compression ability while minimizing compression time. Therefore, the next performance metric used to evaluate the algorithm was its speed of compression and decompression. The system was tested using a Pentium IV machine with 512 MB RAM. Table 3.3 shows the average compression and decompression time obtained for all selected 10 quality factors for the selected images. The same test was also conducted using the standard technique and is included in the table for comparison.
Figure 3.15: Compression Ratio for Different Quality Factors

Figure 3.16: Comparison with JPEG
### TABLE 3.3

**COMPRESSION AND DECOMPRESSSION TIME**

<table>
<thead>
<tr>
<th>Image Name</th>
<th>Proposed system</th>
<th>JPEG 2000</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>CT</td>
<td>DT</td>
</tr>
<tr>
<td>Lena</td>
<td>0.13</td>
<td>0.26</td>
</tr>
<tr>
<td>Pepper</td>
<td>0.43</td>
<td>0.25</td>
</tr>
<tr>
<td>Zelda</td>
<td>1.66</td>
<td>0.09</td>
</tr>
<tr>
<td>Barbara</td>
<td>0.44</td>
<td>0.26</td>
</tr>
</tbody>
</table>

CT – Compression Time; DT – Decompression Time (in seconds)

The compression time taken by the proposed system and JPEG algorithms is 0.66 seconds and 0.71 seconds respectively, while decompression time is 0.215 seconds and 0.225 seconds respectively. Thus, the total average compression time for the proposed algorithm was 0.437 seconds and for the JPEG algorithm is 0.467 seconds. This clearly indicates that the compression time of the proposed approach is less by 7.04% and the decompression time is less by 4.4%.

Figure 3.17 depicts the total time for the proposed and JPEG techniques. The total time was arrived as the sum of compression and decompression times.

According to [76] compression ratio and speed are the two most important performance factors of any compression algorithm. From the figure, it is evident that the speed of compression and decompression in the proposed method is marginally faster when compared to the standard algorithm and therefore makes it an attractive option for several advanced applications like World Wide Web and wireless networks.
c) Peak Signal to Noise Ratio

To measure the quality of the compressed image, the PSNR is calculated. The behaviour of the proposed system and the JPEG algorithm with respect to PSNR is shown in Table 3.4 and Figure 3.18.

**TABLE 3.4**

<table>
<thead>
<tr>
<th>PEAK SIGNAL TO NOISE RATIO in dB</th>
</tr>
</thead>
<tbody>
<tr>
<td>Quality Factors</td>
</tr>
<tr>
<td>-----------------</td>
</tr>
<tr>
<td><strong>LENA</strong></td>
</tr>
<tr>
<td>Proposed System</td>
</tr>
<tr>
<td>JPEG 2000</td>
</tr>
<tr>
<td><strong>PEPPER</strong></td>
</tr>
<tr>
<td>Proposed System</td>
</tr>
<tr>
<td><strong>ZELDA</strong></td>
</tr>
<tr>
<td>Proposed System</td>
</tr>
<tr>
<td>JPEG 2000</td>
</tr>
<tr>
<td><strong>BARBARA</strong></td>
</tr>
<tr>
<td>Proposed System</td>
</tr>
<tr>
<td>JPEG 2000</td>
</tr>
</tbody>
</table>

97
Figure 3.17: Compression and Decompression Time
Figure 3.18: PEAK SIGNAL TO NOISE RATIO
The Table reveals that the proposed system produced high PSNR value indicating that the decompressed image is very near to its original counterpart. The PSNR value varied between the range of 23.90 and 36.46 dB. On an average the proposed system produced a 28.60 dB PSNR value, while it was 27.96 dB by JPEG algorithm and showed a percentage increase of 2.2%. This clearly indicates that the output quality of the proposed system is better than JPEG algorithm and the high PSNR and compression rates clearly indicate that the proposed algorithm achieves good quality compression and produces an image with reduced noise.

According to [113], an improved compression algorithm is recognized by a high PSNR or a lower MSE. In agreement with this, the results of the proposed system with high PSNR prove that it is an improved version over existing methods.

All the tests produced quality images with good visual clarity as is evident from Table 3.2. Decompression results for Lena, Pepper, Zelda and Barbara with varying quality factors (20, 50, 70 and 100) for the proposed approach and JPEG algorithm are shown in Figure 3.19, 3.20, 3.21 and 3.22 respectively.

d) PSNR and Compression Ratio

A comparative analysis was also made between the PSNR and the corresponding compression ratios. The results of the proposed system and JPEG 2000 for Zelda image are schematically presented in Figure 3.23.

From the pattern obtained, it is evident that the PSNR and compression ratio are inversely related (i.e.) the PSNR decreases with the increase in compression ratio. It is also evident that the proposed system is better than JPEG 2000.
Figure 3.19: Lena – Visual Comparison

PSNR : 25.73; CR : 51.83
QF : 20

PSNR : 24.15; CR : 48.93
QF : 20

PSNR : 28.46; CR : 36.98
QF : 70

PSNR : 27.94; CR : 33.76
QF : 70

Original Image

Proposed

JPEG
Figure 3.20: Pepper - Visual Comparison

Original Image

Proposed

QF : 20
PSNR : 24.92; CR : 46.90

QF : 70
PSNR : 28.64; CR : 33.51

JPEG

QF : 20
PSNR : 25.44; CR : 43.12

QF : 70
PSNR : 28.41; CR : 30.95

Figure 3.20: Pepper - Visual Comparison
Figure 3.21: Zelda – Visual Comparison

Original Image

Proposed

JPEG

QF : 20
PSNR : 27.02; CR : 54.81

QF : 20
PSNR : 27.00; CR : 53.26

QF : 70
PSNR : 30.67; CR : 42.86

QF : 70
PSNR : 29.82; CR : 40.20

Figure 3.21: Zelda – Visual Comparison
Figure 3.22: Barbara – Visual Comparison
Figure 3.23: Compression Ratio and PSNR
Thus the various results of the experiments conducted clearly indicate that the images produced by the proposed compression algorithm are of good visual quality with increased compression ratio and therefore can be applied to most of the image processing systems.

3.5 SUMMARY

In this work, a color image compression method based on wavelets is proposed, where YUV color space is employed. The system was developed using Visual C++ and was tested on Pentium IV machine with 512 MB RAM. Experimental results show that the proposed algorithm is easy to implement and moreover it reduces the computation time required. The algorithm yields good compression ratio and high PSNR values for the decompressed images. This shows that the reconstructed images are very close to the original images in visual quality. Future research direction is planned in the use of wavelet packets to improve the performance. The use of different structuring elements and prediction methods that further exploit the redundancy in the color bands to improve the performance may also be considered.

The next chapter discusses and proposes a technique that uses wavelet packets for compressing natural/photographic color images.